UCLouvain

École polytechnique de Louvain

Deep Learning Methods to Enhance Adventure Films

In collaboration with Agile Birds

Authors: **Antoine Caytan, Gauthier de Moffarts d'Houchenée**
Supervisor: **Pierre Dupont**
Readers: **Pierre Dupont, Dominique Snyers, Christophe De Vleeschouwer**
Academic year 2021 – 2022
Master [120] in Computer Science and Engineering

Acknowledgements

First and foremost, we would like to express our gratitude to our supervisor Prof. Pierre Dupont for his guidance and his precious feedbacks.

It is also important to thank Mr Dominique Snyers for his confidence and the images he provided from Agile Birds that helped us throughout the experimental process of this thesis.

Then, we are grateful to the OpenMMLab team for the excellent libraries they openly and publicly release. We would particularly like to thank PhD candidate Mr. Kelvin C.K. Chan and his colleagues for the large amount of work they put into proposing the BasicVSR models family and the few insights he was able to give us.

Let us not forget the computational resources that have been provided by the supercomputing facilities of the Université catholique de Louvain (CISM/UCL) and the Consortium des Équipements de Calcul Intensif en Fédération Wallonie Bruxelles (CÉCI) funded by the Fond de la Recherche Scientifique de Belgique (F.R.S.-FNRS) under convention 2.5020.11 and by the Walloon Region[1]. Without them, the completion of this thesis would have been impossible.

We would also like to thank in advance our readers, Prof. Pierre Dupont, Prof. Christophe De Vleeschouwer and Mr. Dominique Snyers, for their time and their undoubtedly very constructive evaluation.

And last but not least, we find it important to thank our friends, more precisely, Marine Branders and Valentin Lemaire for their help and precious advice.

[1] `http://www.ceci-hpc.be/clusters.html#dragon2`

Abstract

In recent years, the fields of computer vision and image processing have been revolutionized by machine learning techniques and in particular by deep learning models. Indeed, a lot of image processing methods have been rediscovered, even surpassed, by these new artificial intelligence models.

The technique that this article will focus on is super-resolution. Its purpose is to increase the number of pixels in an image while improving its visual quality. Whether it is to improve a single image or a video that is a succession of images, super-resolution models have achieved remarkable results.

This work will attempt to use the good methods behind this technology for a restoration task. That is, the aim will no longer be to increase the resolution, but only to improve the visual quality of the images. In particular, the input images will be spatially reduced within the model to recover the original dimensions in the output after the super-resolution operation. With this design, the majority of the computations are performed on low-resolution inputs and at the same time allows to improve performance.

Table of Acronyms

$BVSR$	BasicVSR
$RBVSR$	RealBasicVSR
FPS	Frames Per Second (in a video)
GAN	Generative Adversarial Network
HR	High Resolution Image
HQ	High Quality Image
LQ	Low Quality Image
LR	Low Resolution Image
$LReLu$	Leaky Rectified Linear Unit
$MISR$	Multiple Image Super Resolution
MOS	Mean Opinion Score
MSE	Mean Square Error
$PSNR$	Peak Signal-to-Noise Ratio
$ReLu$	Rectified Linear Unit
$SISR$	Single Image Super Resolution
SR	Super Resolved Image
VSR	Video Super Resolution

Table of Notations

		Order of magnitude
b	Refers to the backward propagation	
B	Batch size	$B \in \{1, 2, 4, 8, 16\}$
c	Number of feature in the encoding	64
c_i	i^{th} frame of the output of the Cleaning Module	
c_i^p	i^{th} frame of the input video cleaned p times	$p \in 1, 2, 3$
d	Downsampling Operator	by 4 in each dimension
f	Refers to the Forward Propagation	
F_b	The Backward Propagation Module in (Real)BasicVSR	
F_c	The Cleaning Module in RealBasicVSR	
F_f	The Forward Propagation Module in (Real)BasicVSR	
F_u	The Upsampling Module in (Real)BasicVSR	upsample by 4
h	Height of low resolution input images in number of pixels	$h \in [180, 540]$
h_i^b	Output of the i^{th} backward propagation branch F_b	
h_i^f	Output of the i^{th} forward propagation branch F_f	
HQ_i	i^{th} frame of the high quality video	usually 4K resolution
i	Refers to the image index	
j	Refers to the pixel index	
k	Index of a layer $\in K$	from 2 to 34
K	Set of layer indexes used by λ_{per}	
l	Maximum length of a sub-sequence in GPU	usually $l \in [5, 20]$
l_k	Loss $\mathcal{L}2$ applied to a layer $\in K$ of ϕ	
L	The number of frames that compose the input video	$L \in [369, 7500]$
$\mathcal{L}_{1^{st}}$	Loss used for the 1^{st} part of the training of RealBasicVSR	
$\mathcal{L}_{2^{nd}}$	Loss used for the 2^{nd} part of the training of RealBasicVSR	
$\mathcal{L}_{adv}$	Adversarial Loss	
$\mathcal{L}_{clean}$	Cleaning Loss	
$\mathcal{L}_{out}$	Output Fidelity Loss	
$\mathcal{L}_{per}$	Perceptual Loss	
LQ_i	i^{th} Frame of the low quality video	usually 4K resolution
LR_i	i^{th} Frame of the low resolution video	usually 960×540
m	Refers to the sub-sequence index	$m \in [0, L//l]$
MAX_z	Highest possible pixel value in z $2^{\#bits\_encoding\_pixel} - 1$	usually 255
N	Number of pixels in the image	$8,294,400$ for 4K image
N_k	Number of element of the tensor of the encoding at the k^{th} layer of the Loss Network ϕ	

		Order of magnitude
n_{res}	Number of Residual Blocks in the Residual Module	20 or 30
n_{clean}	Number of Residual Blocks in the Cleaning Module	20 by default
p_i	Degradation Parameters of the i^{th} frame in RealBasicVSR	
q	Maximum length of a sub-sequence in main memory	usually 10
r	Residual Bloc	
R	Residual Module	
s	Scaling Factor	4
S	Optical Flow Module	
s_i^b	Backward optical flow at level i in BasicVSR	
s_i^f	Forward optical flow at level i in BasicVSR	
SR_i	i^{th} Frame of the super-resolved video	
$SR_{i,j}$	The j^{th} super-resolved frame of the i^{th} sub-sequence	
SR_i^m	i^{th} output frame of the m^{th} sub-sequence	
T_{cur}	Number of steps since the last restart	$T_{cur} \in [0, T_t[$
T_t	Number of steps between two restarts	400000
w	The width of the low resolution image in number of pixels	$w \in [318, 960]$
W	Warping Module	
w_i^b	Warped backward features at level i in BasicVSR	
w_i^f	Warped forward features at level i in BasicVSR	
z_i	i^{th} Frame of the ground truth video	usually 4K resolution
$\hat{z}$	Reconstructed Image	
Δp_{i+1}	Differences between parameters of the degradation applied on input frames of RealBasicVSR from frame i to frame $i+1$	
ϵ	Parameter of the Charbonnier loss	1×10^{-8}
η_{max}	Maximum Learning Rate	$\eta_{max} \in [1e-4, 5e-5]$
η_{min}	Minimum Learning Rate	
η_t	Learning Rate at the step t since the last restart	
λ_{adv}	Weight applied to $\mathcal{L}_{adv}$ in $\mathcal{L}_{2^{nd}}$	5×10^{-2}
λ_k	Weight applied to l_k in $\mathcal{L}_{per}$	0.1 and 1
λ_{per}	Weight applied to $\mathcal{L}_{per}$ in $\mathcal{L}_{2^{nd}}$	1
ϕ	Loss Network used for λ_{per}	
ϕ_k	Activation of the k^{th} layer of the Loss Network ϕ	
θ	Stopping Criteria of the Dynamic Refinement	1.5, 5 or 255

Contents

Chapter 1

Introduction

In the heart of the taiga, Dominique Snyers, a former machine learning engineer, now founder and director of Agile Birds, spent part of the winter with the last reindeer herders in Mongolia. His latest film, “The Winter Guests”, tells the story of this adventure with images full of emotions. However, the difficult conditions do not always allow obtaining images of sufficient quality. To help him realize his film, our job was to use artificial intelligence models to improve the quality of these videos.

In particular, as the name of the thesis indicates, our work consisted of using deep learning methods to improve the quality of images in adventure films. By "adventure film" is meant a documentary film shot in difficult conditions, whether in low light, with a lot of vibrations or shot in a hurry with non-professional cameras.

In collaboration with Agile Birds, we narrowed the problem down to certain scenes. The vast majority of the rushes are actually of good quality, shot with good cameras, on tripods and in daylight. The problematic parts consist of 5 sequences of about 2 minutes which suffer from some undesirable effects due to low light conditions. Indeed, they were all shot in a tepee, where Mr Snyers and his colleagues spent the night, with the wood stove as the only source of light.

Since a video is nothing more than a succession of images, the early work was about exploring which deep learning techniques could improve the quality of an image. This involved a large period of in-depth research, during which we familiarized ourselves with all the literature on the subject. Because it was so vast, this research was structured around many concepts and techniques, each aimed at improving a particular aspect of images. But one technique stood out from the rest: super-resolution, and in particular video super-resolution.

To perfectly get to grips with this technology, much of the work has focused on

an existing model. It is BasicVSR and its version adapted to real images Real-BasicVSR, proposed by Chan et al. [8] [10] that we decided to study. Being an exploding field of research, these models want to rethink what should be considered as the basic elements of video super resolution in terms of result/computation ratio. This was then deconstructed from scratch in order to understand each of the notions behind its design and to finally be able to highlight some aspects that we could work on.

Naturally, the next step was to propose and implement improvements following this analysis. This was based on three main motivations. The first was the desire to keep the model as light and efficient as possible in order to remain consistent with the chosen model. The second was the desire to keep part of the model intact in order to keep the weights obtained by a training well beyond our capacities. The last motivation was to adapt the model to our problem, which is not a classical super-resolution problem.

In response to this, an architecture was proposed. This one, having been chosen in order to answer the best to our problem, still has some arbitrary choices. To justify this, an ablation study was carried out. This consists of varying these subjectively determined parameters to observe the variation in the performance of the implementation.

Finally, the results obtained from this study were compared and analysed to determine the model that was considered as final. This model was then fine-tuned with a dataset created from the good quality images provided by Agile Birds, in an attempt to further improve its performance.

Chapter 2

State-of-the-Art

2.1 Introduction

Before getting to the heart of the work, it is interesting to get an idea of the progress of research in the field around which this paper is based, namely artificial intelligence in the field of computer vision.

2.1.1 Terminology

Foremost, to make sure that we have a good foundation, let's quickly go over some starting concepts and their terminology.
Artificial intelligence refers to programs that make decisions by themselves. This has several subsets, one of which is machine learning. In this paradigm, the program still makes decisions autonomously, but this time based on a model that has parameters that have been trained on data. Finally, machine learning itself has a branch called Deep Learning. This is machine learning, where the models are deep (in terms of the number of layers) and take raw data as input rather than handcrafted features. It is of course the latter branch on which this work will focus.

2.1.2 Tasks

The general principle of the application of artificial intelligence in the field of computer vision is to collect pixel-wise features based on topology, geometry to be able to perform certain operations on images. A lot of research manipulating images have taken place in the past and yielded more than satisfactory results. Here is a small overview of some of the tasks that have been and are still being addressed.

- **Image-to-image Translation** :Translating one possible representation of a scene into another [17]
- **Image Inpainting** : Filling holes with plausible content in images
- **Image Dehazing** : Removing the haze from an image.
- **Super Resolution** : Generating high-resolution images (SR) from low resolution ones (LR).

The task that will be at the heart of the subject in this document is super resolution.

2.2 Super Resolution

2.2.1 Problem Statement

Super-resolution refers to the process of improving the spatial resolution of an image. That is, algorithmic techniques that, from one or more images, create an image of higher resolution (that has more pixels).

In general, the super-resolution problem is formulated as recreating a high-resolution image HR from one or more low-resolution images LR_1, ..., LR_L. The term resolution should be understood here as the level of detail, i.e. the level of significant high frequencies contained in the image. Generally, the result produced by the algorithm has larger dimensions than the input image, i.e. a better definition in terms of pixels. The scaling factor s, will be noted hereafter, which is nothing more than the ratio between the input and the output sizes. Sometimes super-resolution algorithms are used to improve the quality of an image without enlarging it.

2.2.2 Methods

Different approaches have been developed to solve the super-resolution problem.

Interpolation

Simple interpolation algorithms can be used to obtain a larger image. Although they are very fast, due to the simplicity of the calculations they implement, they often provide results that are too smooth, i.e. they fail to recreate high frequencies (edges, corners). Sometimes they are used to process only sub-parts of an image that contain fewer high frequencies, such as a blue sky. Other parts containing high frequencies are then processed with a more sophisticated but slower method.

Several interpolation methods exist, but they all work in the same way. As shown in Figure 2.1, the pixels are spaced apart and the gaps are filled by the interpolation technique used. Let's take one of the best known, the bilinear interpolation. It consists of taking the up to 4 values surrounding the desired point and applying 2 classical linear interpolations, one vertical and one horizontal.

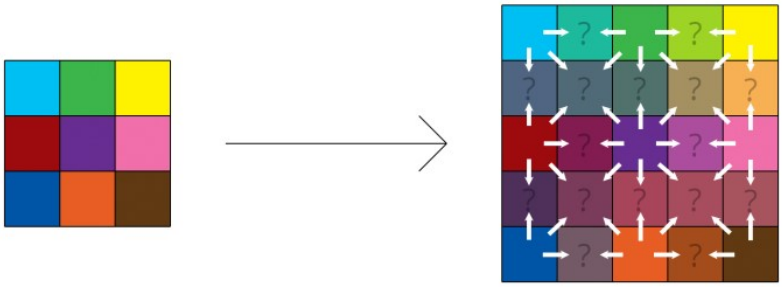


Figure 2.1: Illustration of the Bilinear Interpolation Method. The arrows represent which pixels were used to calculate the value of the unknown pixels. [16]

Tile methods

Other methods are based on what is called tiles. These aim to learn the similarities between low and high resolution tiles using training images. By learning a dictionary of high and low resolution tiles for a certain scaling factor, it is possible to calculate the high resolution version of an input image using the high resolution tiles equivalent to the low resolution tiles that make up the input image. Sometimes this linear combination of tiles is coupled with certain constraints to avoid edge effects.

Learning Based Approaches

All of the above methods have been overtaken by the emergence of learning based methods. It consists of approximating a mapping between a low-resolution image and a high-resolution image using machine learning tools.

Learning-based super-resolution methods can be distinguished into 3 main branches. The first is Single Image Super Resolution (SISR). This is when the problem is to enhance a single image with itself as the only input source. This is the case of an old photo taken with a bad camera, whose resolution is enhanced. Now, if several relatively identical shots exist, they can be used together to get more information but still get a single enhanced image. This is called Multiple Image Super Resolution (MISR). Finally, in the case we are interested in, a film is a succession of highly similar images with sometimes only a small translation between

the shots that are represented. There is then Video Super Resolution (VSR), which, depending on the model, uses all (or almost all) of the images that make up the shot that is to be enhanced. In practice, each frame of the shot is successively enhanced using the previous and/or following input frames.

Deep Learning based Approaches

Among the techniques whose approach is based on learning, those that use deep learning have obtained results beyond what has been done.

In 2015, Dong et al. [12] proposed a convolutional neural network called SRCNN. This network is composed of 3 layers and is guided by a loss function based on a mean square error. This method became the State-of-the-Art at that time and a reference for deep learning methods. Since then, a common structure that governs deep learning super-resolution methods has been identified in the publication of Wang et al. [33].

The first axis around which a model is built is what is called the *Model Framework*. This is the position of the upsampling calculation in the model. The low-resolution image is scaled up to the size of the high-resolution image. There are various possibilities, but the three main ones are :

- **Pre-upsampling** : The upsampling is done at the beginning of the model
- **Post-upsampling** : The upsampling is an integral part of the network and is located in the last layers of the convolutional network.
- **Progressive-upsampling** : The upsampling is done along the whole model.

Once the position of the upsampling has been defined, the next step is to determine the *upsampling method* that is used. The method used can be done either by an interpolation method (bilinear, bicubic, nearest neighbour, ...), or by a learning method generally in the form of network layers such as convolutions, pixelhuffles and so on. Secondly, the overall *design of the network* can be different from one model to another. Among the main designs, there are :

- **Residual** : Utilize skip connections to let the network learn the residue between the input and the output instead of the complete mapping.
- **Recursive** : Apply the same set of weights recursively over a structured input.

- **Dense** : Utilize layers whose inside neurons connect to every neuron in the preceding layer.

Finally, the network learning must be guided by a *loss function*. The best known class of losses are the pixel losses, which compare the value of pixels directly with each other. Another family is content loss, which is calculated on the encoding of the images. That allows the loss to focus on the content of an image without being constrained to an exact correspondence between the generated image and the real image.

Other improvements can be made to the model such as data augmentation, multitask learning, etc.

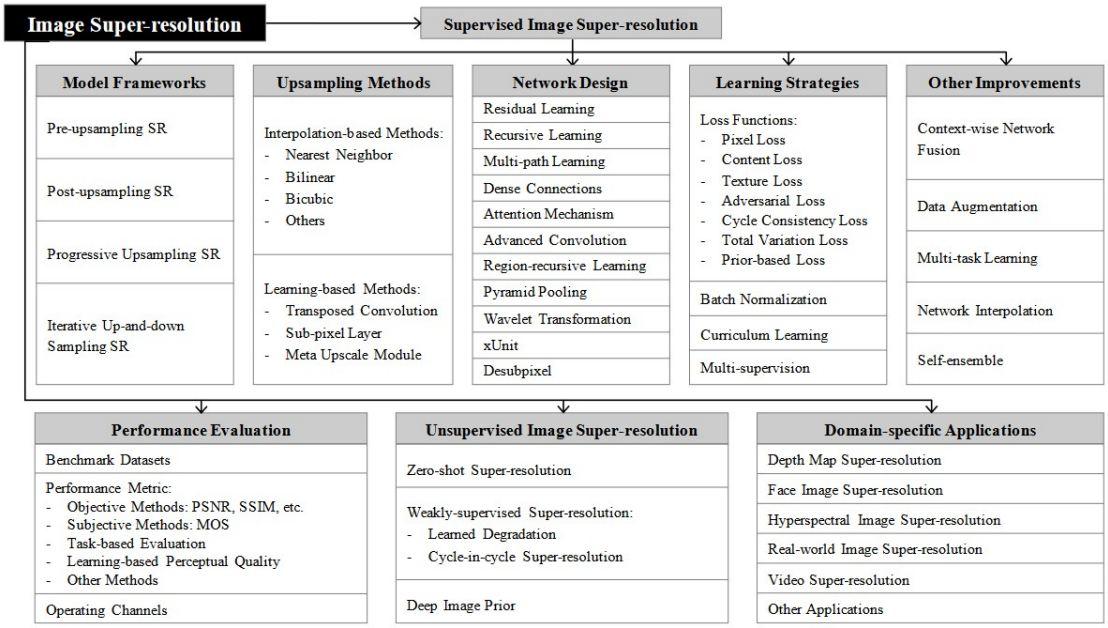


Figure 2.2: Super Resolution Hierarchy found by Wang et al. [33]

Beyond this common structure, in 2014, Goodfellow et al. [13] introduced a new class of unsupervised learning algorithm called Generative Adversarial Network (GAN). Such an architecture consists of 2 networks that compete against each other in the form of a zero-sum game where one agent's gain is another agent's loss.

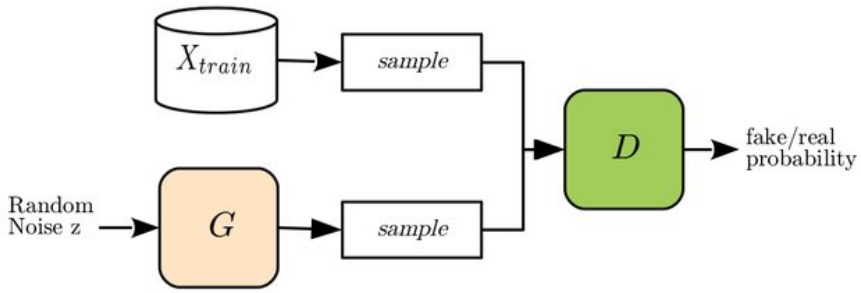


Figure 2.3: Illustration of a Generative Adversarial Network by Hayes et al. [14]

Starting from a given dataset, the first network is said to be generative, and its goal is to generate images of the same kind as the dataset (with the same statistics). The second model, on the other hand, is called a discriminator, and tries to determine whether an image it receives is real (belonging to the dataset) or whether it is an image generated by the generator.

The training of this architecture is based on a known dataset that will be used initially for the discriminator. The training consists of presenting to the discriminator, images from the dataset X following the probability $P(X)$ or images generated by the generator based on noise drawn from the prior distribution $P(Z)$. By doing so, the generator then trains itself according to its ability to fool the discriminator with the quality of the generated images. More precisely, the generator G and the discriminator D are competing in a minimax game to optimize an objective value :

$$min_G max_D[\mathbb{E}_{X\sim P(X)}[log(D(X))] + \mathbb{E}_{Z\sim P(Z)}[log(1 - D(G(Z)))]] \tag{2.1}$$

Indeed, the discriminator D evaluates its input by returning a value between $[0, 1]$. A value close to 1 means that the image is evaluated as real (belonging to the dataset) and 0 if the image is evaluated as fake (generated by the discriminator).

Independent back-propagation procedures are applied to both networks so that the generator produces better samples, while the discriminator becomes more accurate at reporting fake samples.

In the end, the aim of such an architecture is simply to extract the generator to obtain a model capable of generating images.

In fact, GANs and more precisely cGANs have been a small revolution in the field of super resolution, especially in SISR. cGAN, which stands for conditional GAN, is a variant of GAN that, instead of having a generator capable of generating

images from noise, will take another image as input. In the case of super resolution, the cGAN generator will take a LR image as input and will learn to generate SR images, while the discriminator will have to determine whether an image is really high resolution or whether it has been super-resolved.

2.2.3 Measurement methods

It is hard for a computer to assess if an image looks real to a human eye or not. Using a mean square error (MSE) which would be a simple pixel-wise difference between a produced image and the ground truth image, does not emphasize the details that are relevant to humans [35]. It encourages pixel-wise averages that lead to overly-smooth solutions [5]. To answer this, 2 main families of solutions exist, the objective methods, and the subjective methods, which require a human evaluation.

Although the idea sounds very appealing, subjective methods are generally very complicated to implement. This is the case of the mean opinion score (MOS) which consist of asking humans to rate the quality of images on some particular scale (from 1 to 5 where 1 is bad and 5 is excellent).

Therefore, the majority of the methods used are objective. The question that then comes to mind is what aspects are important in assessing the quality of an image? In fact, two big aspects come into play when analysing how the quality of an image is assessed, namely, distortion and perception. The *distortion* is defined as the dissimilarity between the reconstructed image and the original one. The *perception* refers to the visual quality of the constructed image[3].

In fact, it is important to notice that a trade-off exists between perception and distortion. Blau et al.[3] proved that there exists an unreachable region in the perception-distortion plane. It means that if an algorithm is close to the bound defined by Blau et al., it is not possible to improve both the perceptual quality and the distortion. Then, a choice of design has to be made. In some applications, one would prefer to favour the perceptual quality over the similarity to a reference image. For example, in medical imaging, the reconstruction accuracy, and thus the distortion might be preferred to the visual quality, the perception, to avoid seeing something that does not exist in reality.

Distortion Metrics

Blau et al. cite several methods that measure the distortion [3]. Among those, the methods commonly used are the MSE and the peak-signal-to-noise ratio (PSNR)

whose formulas are linked.

$$MSE(z, \hat{z}) = \frac{1}{N} \sum_{j=0}^{N} (\hat{z}_j - z_j)^2 \tag{2.2}$$

where N is the number of pixels, z is the original image and $\hat{z}$ the reconstructed image. The j refers to the pixels indexes.

The PSNR is then a way to measure the error relative to the intensity of the signal based on the Mean Square Error.

$$PSNR = 10 \cdot log_{10} \left(\frac{MAX_z^2}{MSE(z, \hat{z})} \right) \tag{2.3}$$

Here, MAX_z is the highest possible value assignable to a pixel in z.

Perception Metrics

The Mean Opinion Score (MOS) described above, in addition to being subjective, is also a perceptual metric. Human intervention in the metrics permits to avoid looking for the elements that make an image perceptually good. Only to avoid having to deal with an objective evaluation at every inference, datasets of human evaluated images have been created to be reused. This way, researchers have developed algorithms capable of measuring the perceptual quality of an image based on its statistics and its deviations from the statistics of natural images. This is the case of the No-Reference Quality Metric (Ma) proposed by Ma et al. [20].

Ma is a perceptual evaluation method that is based on a model that has been trained on images whose quality has been rated by humans. Without going into too much detail (because it is difficult to do so without understanding many concepts that are not covered here), a model has been trained to find the mapping between features of the images and the rating received by the subjective evaluation.
The first feature analysed by the model is about the frequencies of the image. Indeed, since super-resolved images are generated from low resolution inputs, the task is equivalent to restoring high frequency components to LR images. To quantify these high frequencies introduced by the restoration, we use the coefficients obtained by transforming the images into the discrete cosine domain.
The second feature is the spatial discontinuity of the pixel intensity. This is done by analysing the average intensity of groups of pixels within an image.
In this way, an image is evaluated by inferring the model on that image. The result is a score equivalent to the one given by humans. In practice, the higher the score,

the better the quality.

Other metrics attempt to completely detach themselves from human intervention in perceptual assessment. This is the case of the Natural Image Quality Evaluator (NIQE) [24].

Indeed, this metric does not require any human intervention, as it only uses statistical features present in natural images. Again, a model is used to implement this metric, in this case a Multivariate Gaussian model (MVG). In practice, natural images (taken from Berkeley image segmentation database [21] and from copyright free Flickr data) are analysed for several features and the distribution of the values of these features will be used to fit the MVG model. Then, when a new image have to be evaluated, it is analysed on the same features. With this data, another MVG model is fitted. The difference between the two models is then used to quantify the quality of the evaluated image. More precisely, the distance between the two MVG is calculated as follows :

$$D(\nu_1, \nu_2, \Sigma_1, \Sigma_2) = \sqrt{\left((\nu_1 - \nu_2)^T \left(\frac{\Sigma_1 + \Sigma_2}{2}\right)^{-1} (\nu_1 - \nu_2)\right)} \tag{2.4}$$

Where ν_1, ν_2 and Σ_1, Σ_2 are the mean vectors and covariance matrices of the natural model and the tested image's model. Unlike the Ma, the lower the score, the higher the quality.

Another completely different method of avoiding human intervention is based on the Generative Adversarial Networks. Indeed, while the generator is used for inference, the discriminator can very well be used for learning. In fact, it is possible to couple the output of the discriminator to a loss function to obtain a value that have to be minimized. More precisely, the output of a super-resolution model can be sent to a trained discriminator. The discriminator will then evaluate whether the images it receives as input are generated images or real ones by returning a continuous value between 0 and 1 respectively. Then this prediction will be compared with the real image binary labels (0 or 1) using the Binary Cross Entropy given by the following equation :

$$BCE = -\frac{1}{B}\sum_{i=0}^{B}[y_i \cdot log(\hat{y}_i) + (1 - y_i) \cdot log(1 - \hat{y}_i)] \tag{2.5}$$

where y_i is the true label of the i^{th} image, i.e. if the images comes from a dataset or if it is generated. $\hat{y}_i$ is the predicted label of the i^{th} image. B refers to the batch size.

This loss function is typically appropriate when there are only 2 labels. Notice that when the real label is 1, the second half of the function disappears. In the case where the real label is 0, the first half of the equation disappears. In short, we simply multiply the logarithm of the predicted real probability for the ground truth class.

Other metrics that compares in a perceptual manner the quality of images exist. One could cite the Blind/Referenceless Image Spatial Quality Evaluator (BRISQUE) [23], the Learned Perceptual Image Patch Similarity (LPIPS) [36], the Fréchet Inception Distance (FID) [15], the Kernel Inception Distance (KID) [1] and the Video Multimethod Assessment Fusion (VMAF) [4]. However, these metrics will not be detailed in this report.

Chapter 3

Problem Description

At first glance, the problem that this work has to address is the one of improving the quality of adventure films. But, the quality of a film can be improved on many aspects, such as stabilization, the amount of frames per second, the colourization of the images and so on... Only, in the context of this work, not all of these aspects can be solved. Let's have a look at the images that have to be processed to know a little more about what we should focus on.

Agile Birds provided a small database of images and videos from their trip to the Mongolian Taiga. Among them, almost all the rushes are of excellent quality. Having been shot in daylight with good equipment, the images are stable, in high resolution (Full HD or 4K) with at least 25 frames per second. In fact, only a small part of it needs to be reworked. This consists of 5 sequences of about 2 minutes, shot in a teepee where everyone spent the night.

Figure 3.1: Example of lower quality images provided by Agile Birds. A noise effect can be seen despite the high resolution of the image, especially visible in the background and on the faces.

As shown in figure 3.1, the images suffer from a mix between a pixelisation and a

noise effect. This refers to artefacts and unwanted effects that give the image a grainy texture, with randomly coloured pixels interrupting the sharpness of smooth areas. This is a phenomenon that occurs when high resolution images are captured in low light conditions. High resolution increases the level of detail in photographs, and meets professional requirements. But this advantage has its limits, especially for low-light exposures. Increasing the number of pixels on a sensor of unchanged dimensions reduces its dynamic range, which can increase the relative importance of noise compared to the useful signal. In fact, this is the only thing the images suffer from.

From then on, the problem becomes twofold. Deep learning methods will have to reduce the noise as much as possible while restoring details of the images.

Chapter 4

Solution

What is important to realize is that the problem we are facing is not exactly the same as a classic super-resolution problem. In general, super-resolution is applied to low resolution images in order to increase the number of pixels in the image and hopefully improve the visual quality. But, in this work, both the rushes to be improved and the rest of the data are already of excellent resolution, namely Full HD (1920×1080) or even 4K (3840×2160).

In fact, what we are looking to do is not to increase the resolution of our images, but to use the methods behind super resolution to improve the quality of our images. That's why the idea behind all this work will be to reorient the use of super-resolution methods to generalize it to a restoration tool.

To begin, this section explains the first major part of the work, which was to choose an existing model capable of super-resolution and run it on the images provided to us. The aim is to deconstruct this model to understand every detail of its functioning, and thus to be able to highlight its pros and cons. This will lead to the next chapter, which will focus on the contributions that can be made to the model.

4.1 BasicVSR

With the results it obtained, deep learning applied to computer vision, and especially super resolution, is a very fashionable topic. Therefore, a wide range of models exists and are still published today. Despite this, a choice had to be made, and this choice turned to the BasicVSR model introduced by Chan et al. [8]. This is a Video Super Resolution model capable of increasing spatial dimensions by a factor of 4.

In a nutshell, the main reason behind this choice is that this model does not seek to be state of the art. Rather, it wants to rethink what should be considered the basic elements of video super resolution in terms of the result/computation ratio.

To elaborate further, that model is simple. Indeed, the architecture is made up of only a few modules that allow the implementation of the 3 main axes around which this Video Super Resolution model is built : *Propagation*, *Alignment* and *Upsampling*.

Firstly, as you know, a video is actually made of successive images. By nature, two consecutive frames of a video are almost identical, sometimes slightly misaligned. Therefore, it is common to derive information not only from spatial dimensions but also from the temporal dimension. This means that in order to improve one of the images in the video, its neighbours will also be analysed. This is called propagation.

The second axis is the alignment. Now that the neighbouring images are accessible thanks to propagation, they must be analysed correctly. The problem is, as written above, that these frames are slightly shifted from each other. To remedy this, it is possible to deform them in order to match features and/or areas that are highly correlated.

Finally, the third axis is the upsampling. This is simply the function that will increase the spatial dimensions of the images.

A second reason for choosing this model is that it is qualitative, both in terms of quality and efficiency. Indeed, it achieves results at least as good as the state of the art while being efficient in its execution, thanks to the fact that the model has far fewer parameters than the standard.
Moreover, the model is versatile. Its modular structure makes it easy to add, modify or delete some of its building blocks.

4.1.1 Global Architecture

Basically, the model has to increase the resolution of a video. As the video is made of successive images, all the work is to improve these images one by one. The input image will be called Low Resolution (LR) while the enhanced image will be called Super-Resolved (SR). Note that we will use the words frame and image as synonyms.

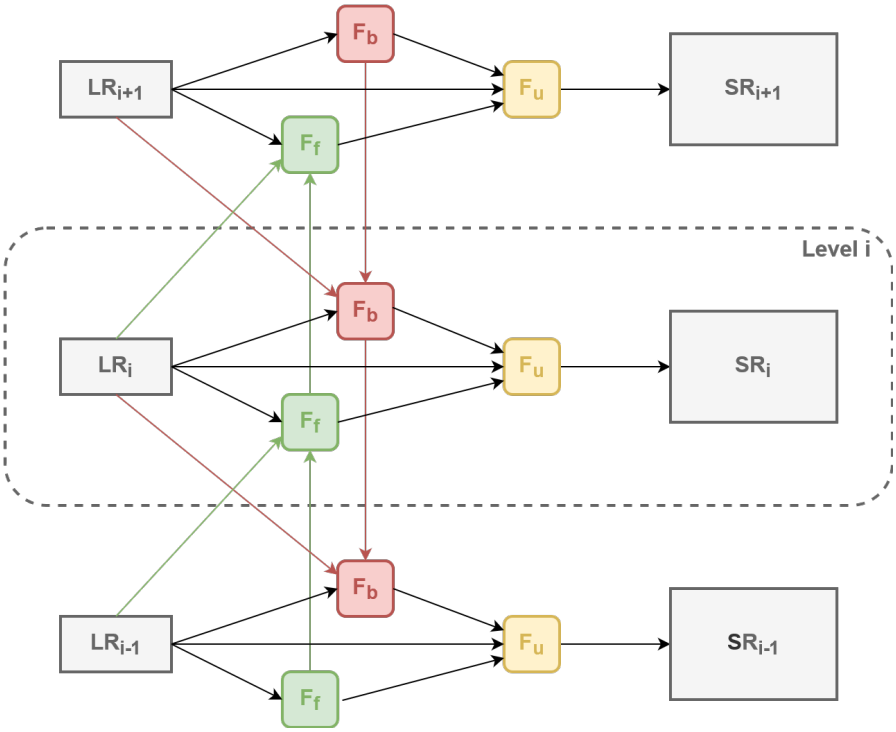


Figure 4.1: BasicVSR Global Architecture

Let's start with a global view of the complete architecture of the model. On the left-hand side of figure 4.1 we can see the different images in the order in which the video is displayed. This means that at the bottom, there would be the first image of the video and at the top the last one. Here we show only 3 consecutive images with LR_i that stands for the i^{th} frame of the low resolution video. On the right you can see the same square shape but this time larger, which represents the enhanced image that will constitute the high resolution output video. SR_i represents the i^{th} frame of the super resolved output video. In practice, the model is build in a recurrent fashion. Indeed, its core architecture is in fact a single level. But, this level is repeated as many times as there are images in the video.

In the centre of the diagram there are 2 vertical branches which are the 2 branches that will enable the information to propagate in time. In green, the forward branch that propagates information forwards, from a level i to the next level $i+1$. In red, the backward branch that propagates information backwards, from a level i to the previous level $i-1$. Note that the colour of the arrows serves as a reminder. A green arrow means that it is part of the forward propagation. Whereas a red arrow is part of the backwards propagation.

Finally, the upsampling module, in yellow, does not propagate any information. It simply collects the low resolution image of the current level (LR_i), as well as the information returned by the forward and backward propagation modules, in order to calculate the super resolved output image.

4.1.2 Detailed Architecture

Now that the links between the different branches and modules are known, they have to be detailed. To do this, one last notion must be explained, namely, the optical flow.

Optical Flow

The optical flow represents the displacement of the scene between two consecutive images. In practice, we will try to match each pixel of the initial image to the final image and thus associate a displacement vector with it.

Figure 4.2: Example of optical flows calculated from 2 consecutive frames

Figure 4.2 shows 2 examples of optical flows calculated from 2 consecutive images. In the upper example, notice that the desk chair, in the top left corner of the image, has moved upwards between the two frames. The set of pixels that constitute it must therefore be associated with a vertical translation vector. To represent the flow visually, each vector is associated with a colour. In this case, the positive vertical vector with a relatively large amplitude is purple, like you can see in the legend. This is why the desk chair is, indeed, represented in purple on the visual representation of the optical flow.

In practice, the optical flow is a tensor with the shape $(h, w, 2)$ where h is the height and w the width in number of pixels of the image. The 3rd dimension has 2

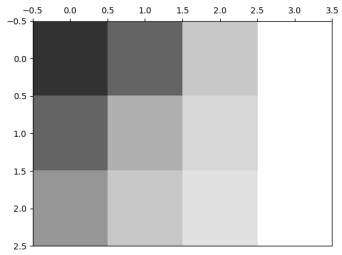
components, since each pixel is associated with a displacement vector $(\Delta x, \Delta y)$.

In BasicVSR the flow is obtained using the pre-trained SPyNet model [28][1]. In practice, the purpose of this flow is not simply to extract motion information between 2 images. In fact, it is often used to distort these images that differ by a small shift to spatially align them. To do this, you need to apply the flow to an image, and this operation deserves a little more explanation.

Let's illustrate our words with a little example. Figure 4.3 shows an example of a gray-scale image and a flow that will be applied to it.

50	100	200	255
100	175	215	255
150	200	225	255

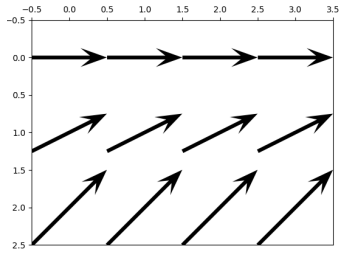
(a) Intensities of a gray-scale image



(b) Gray-scale image of Figure 4.3a

(1, 0)	(1, 0)	(1, 0)	(1, 0)
(1, -0.5)	(1, -0.5)	(1, -0.5)	(1, -0.5)
(1, -1)	(1, -1)	(1, -1)	(1, -1)

(c) A flow represented as $(\Delta x, \Delta y)$



(d) Figure 4.3c represented graphically

Figure 4.3: Example of the application of a flow to an image

Before we begin, note that the conventional image coordinate system is used here. This means that the x axis points to the right of the image, and the y axis points down. In other words, the coordinate $(2, 1)$, represents the pixel of the column index 2 and the element index 1, i.e. the value 215 in Figure 4.3a

The most important thing to understand is that the flow describes the movement of pixels from the initial image to the final image. In other words, a pixel moves from position $(x - \Delta x, y - \Delta y)$ to (x, y). This implies several important things. First, it means that a pixel can be projected to several coordinates by the flow.

[1]Unfortunately, this model will not be developed further in this document.

Secondly, a pixel can take a value that straddles several pixels, in which case the projected value will be a linear interpolation of the straddled values.
Furthermore, it also means that when applying the flow to an image, information may be missing or conversely, information may be sent out of the image.
These two last implications are illustrated below.

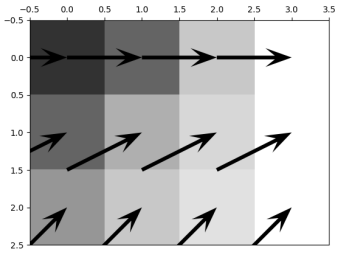
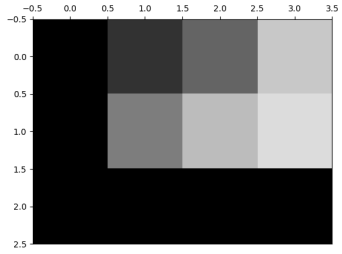


Figure 4.4: Illustration of the application of the warping

Let's take the flow that applies to the first line. It is quite simple since each element in the line is equal, $(1, 0)$ which describes a 1 pixel shift to the right. But then, it means the following. The pixel $(0, 0)$ will receive the information of the coordinate $(-1, 0)$, which we do not have. In this case, we take a null value which will be interpreted by a black pixel and sent to the top left pixel. It doesn't matter what value was at that position before. This can be seen clearly in Figure 4.5 where the whole left side and bottom of the image suffers from this effect.

0	50	100	200
0	125	187.5	220
0	0	0	0

(a) Intensities of the warping of Figure 4.3a by the flow 4.3c



(b) Gray-scale image of Figure 4.5a

Figure 4.5: Result of the application of the warping of Figure 4.3

What happens on the second line is trickier. This time $\Delta y = 0.5$. It means that the value that should be taken by our pixel lies between multiple pixels. Let's take the pixel at the coordinate $(1, 1)$. With a flow equal to $(1, -0.5)$ the pixel, is going to take the intensity present in $(1 - 1, 1 - (-0.5)) = (0, 1.5)$. In this case, the value that will be taken will be calculated using an interpolation of 2 values, here, 100 in $(0, 1)$ and 150 in $(0, 2)$, which gives 125. Note that sometimes the value to take may be just in between 4 pixels, in which case a bilinear interpolation will be required, which is nothing more than a combination of a vertical and a horizontal interpolation.

The Forward/Backward Module

Now that the basics are in place, let's get deeper in the architecture. The first things to detail are the modules that constitute the propagation branches. In fact, when looking more in details, the modules that make up the forward and backward branches are identical. So, let's take one of these modules in the forward branch. For the sake of clarity, figure 4.6 illustrates only one level of the BasicVSR architecture, corresponding to the grey dotted box in figure 4.1.

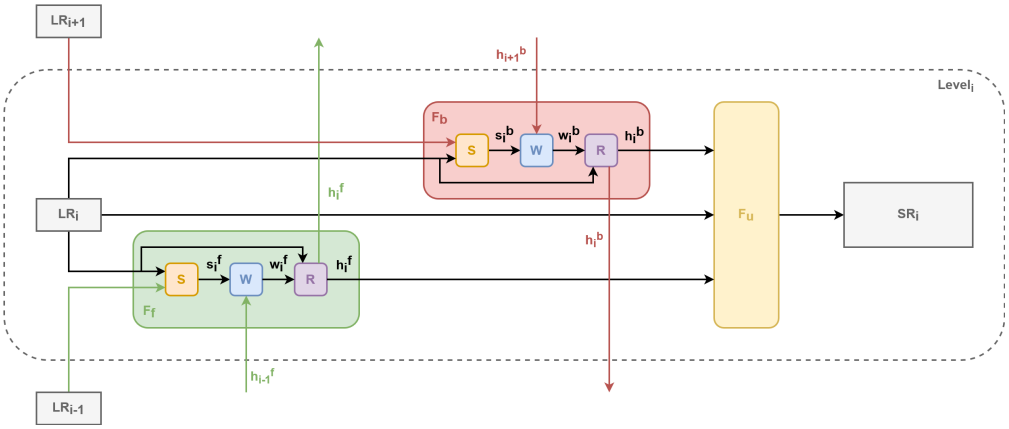


Figure 4.6: Detailed Architecture of one level of BasicVSR

Let's call the module of the forward propagation branch F_f. It consists of three submodules. The first module, in orange, called S, uses the SPyNet model to calculate the optical flow s_i^f from the current low resolution frame LR_i and the previous one LR_{i-1}.

$$s_i^f = S(LR_i, LR_{i-1}) \tag{4.1}$$

The second module, in blue, called W, is a warping[2] module. This is where the flow will be applied like it was explained in section 4.1.2. However, unlike many models [18], [34], BasicVSR has made a different choice. Instead of applying the flow to the raw images, it will be applied to their encoding. In particular, the alignment is done on the output of the previous forward module h_{i-1}^f thanks to the forward propagation connection.

$$w_i^f = W(s_i^f, h_{i-1}^f) \tag{4.2}$$

Finally, the last module, in purple, called R is a residual module. It acts as an encoder of the information it receives as input, in this case, the output of W, called

[2]Warping means to deform, to distort.

w_i^f and the current frame LR_i. This means that the information will be represented (encoded) differently, i.e. with more features. This new representation is richer as it contains information from all previous frames that have propagated so far. The output of this module, denoted h_i^f, is sent to the upsampling module but also to the forward module of the next level. In particular, h_i^f is sent to the warping module of level $i+1$, to be able to do the same work as it is done here.

$$h_i^f = R(w_i^f, LR_i) \tag{4.3}$$

Note that the backward block performs the same operations, only this time comparing the current image with the next image. This means that the warping module receives the current low resolution image LR_i along with the next one LR_{i+1}. The flow extracted from it will be applied to the output of the backward branch of the upper level h_{i+1}^b. Finally, to be encoded and sent to the upsampling module and the lower level. In general, we have

$$\begin{aligned} h_i^f &= F_f(LR_i, LR_{i-1}, h_{i-1}^f) \\ h_i^b &= F_b(LR_i, LR_{i+1}, h_{i+1}^b) \end{aligned} \tag{4.4}$$

where F_f and F_b denote the forward and backward propagation modules, respectively.

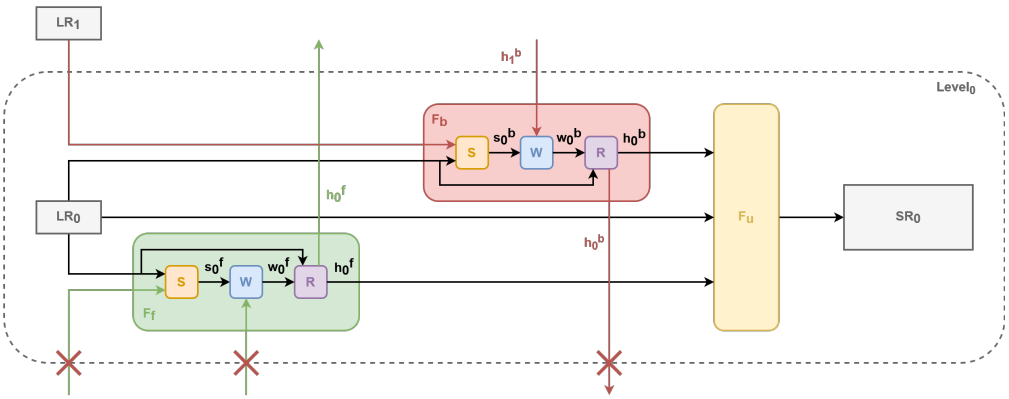


Figure 4.7: Detailed Architecture of the 1st level of BasicVSR

A case that deserves some attention is the borderline case of the first level of the forward branch. This case is different since there is no propagation from the lower level. The S module does not calculate the optical flow since it has no previous frame. Then, the warping module W has no need since its 2 inputs do not exist. In fact, all that is happening here is that the residual module will encode the information of the frame LR_0. This encoding, h_0^f will be sent to the upsampling

modules but also to the forward module of the upper level. Again, this reflection is fully transferable to the backward propagation at the last level.

The Residual Module

As mentioned above, the residual module encodes the information it receives as input. But, let's see more in detail how this is done. Essentially, the residual module is an encapsulation of a residual network. As a reminder, a residual network is formed by the stacking of several residual blocks. A residual block is formed by multiple (usually 2) weighted layers with the particularity of having what is called a "skip connection" that links the input with the output.

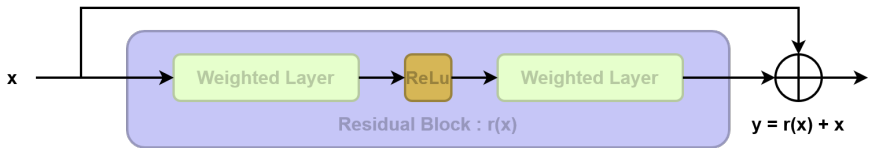


Figure 4.8: Generic Architecture of a Residual Block

Usually, a deep network learns the mapping M from an input x to an output y, i.e.

$$M(x) = y \tag{4.5}$$

Instead of learning a mapping directly, the residual block uses the difference between a mapping applied to x and the original input x, i.e.

$$r(x) = M(x) - x \tag{4.6}$$

With the skip connection used in here, it becomes :

$$M(x) = r(x) + x = y \tag{4.7}$$

The residual block globally tries to learn the true output, M(x), but in fact, since we have an identity connection from x, the layers are actually trying to learn the residual, r(x). Hence, the name: Residual block. The great advantage of a residual network is that it alleviate the vanishing/exploding gradient problem. This is because if the activation of a specific layer tends to 0 in the network, the skip connections will at least transmit what was present before that layer.
It has also been observed that it is easier to learn the residue between the output and the input, rather than just the input [29]. Note that there are several types of residual blocks. The weighted layers can be linked together by different non-linearities and/or normalization functions.

BasicVSR adopts the architecture shown in figure 4.9. It first processes its inputs and then sends them to the residual network itself. In fact, the current image LR_i is concatenated with the output of the warping module w_i [3] and then fed through a convolution, which is used to restore the number of features c. Where c is the parameter which defines the number of features present in the encoding.
Note that, of course here, the convolution serves to fall back on a number of channels equal to c. But more than that, being a learned layer, this one is already learning some shallow features. Then, the residual network is made up by stacking the residual blocks n_{res} times. These are fairly generic since they contain 2 convolutions connected by a Rectified Linear Unit activation function (ReLu). Note the absence of a normalization layer within a block. The removal of a batch normalization layer has been shown to increase performance and reduce computational complexity in various tasks, including super resolution [32].

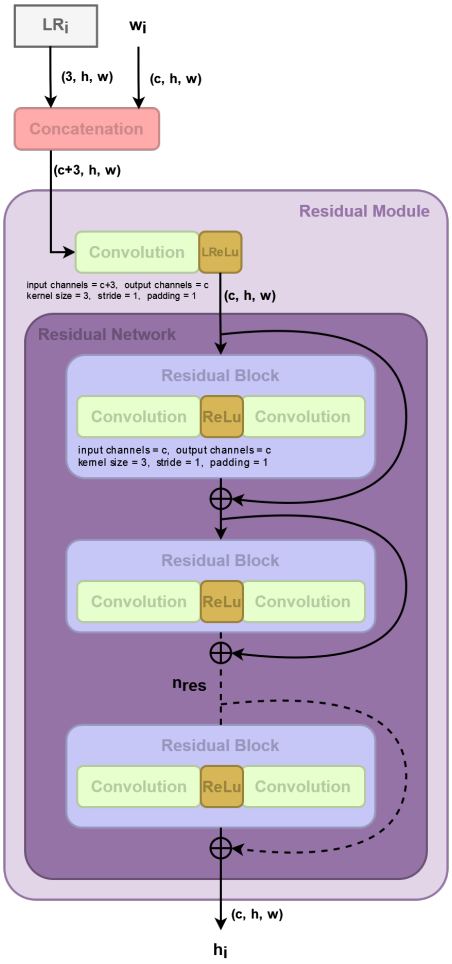


Figure 4.9: Detailed Architecture of the Residual Module

Another feature of this residual network is that it preserves the 3 dimensions of the input. Indeed, neither c is further increased, nor w and h (the spatial dimensions of the signal) are further decreased. In the super-resolution task, we seek to restore the missing details. Therefore, contrary to what is sometimes done in residual networks, the spatial dimensions are not further reduced as they are already at low resolution [7]. In practice, in a classical level such as $level_i$, the entries of the residual block have already the right shapes, i.e. : $LR_i : (3, h, w)$ and $w_i : (c, h, w)$.

[3] The parameter f or b in the exponent of w_i which indicates whether it is the backward or forward module is not written here as the residual module is identical in both cases. This explanation is therefore generic

It is more particular in the limit cases, as for example the $level_0$, again in figure 4.7. There, the entries also have the same shapes. The difference is that, yes, w_i will be a tensor with the shape (c, h, w) but completely filled of zero since there is no previous level.

The Upsampling Module

The upsampling module is an encapsulation of several operations that decode the information it receives as input in order to upscale spatial dimensions by a factor 4. As a reminder, it receives 3 inputs: 2 encoded inputs that are the outputs of the forward and backward modules and a non-encoded input that is the current frame LR_i directly.

Without going into too many details, the upper part of figure 4.10 illustrates that the internal blocks force the number of features of the encoded inputs to 64 and then to 3 in order to find back the dimensions of what should be an image. Convolutions are used to work on the number of features in the signal, while spatial upsampling is done through PixelShufflePack blocks.

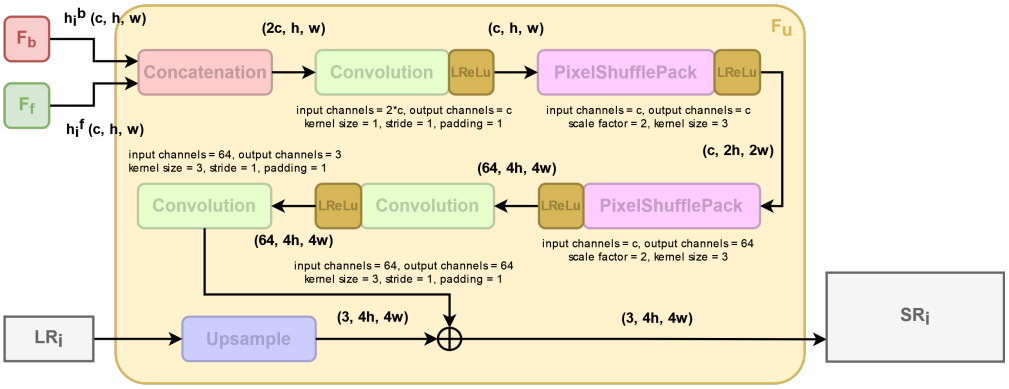


Figure 4.10: Detailed Architecture of the Upsampling Module

A PixelShufflePack is the method used by BasicVSR to increase the dimension thanks to a PixelShuffle operation. A PixelShuffle operation is used to increase the spatial dimension by flattening the channels of the input. As shown on the right of figure 4.11, the s^2 channels will result in a scaling factor of s, both in width and height. On the left side of figure 4.11, the PixelShufflePack simply adds a convolution before the PixelShuffle operation in order to increase the number of channels to $c \times s^2$. In this way, the output of the PixelShufflePack will have the correct number of channels, i.e., c, but upsample these spatial dimensions by a factor of s, in this case, equal to 2.

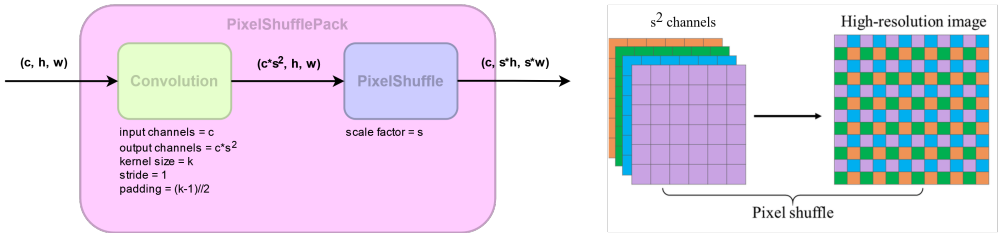


Figure 4.11: Detailed Architecture of the PixelShufflePack with the Illustration of the PixelShuffle Operation [27]

To come back to the architecture of the upsampling module, it is interesting to note the operation that is performed on the current image LR_i. The frame is upsampled by a bilinear upsampling operation. That means that the pixels are moved apart from each other and the values of the "holes" are then interpolated from the known values vertically and horizontally. This interpolation will be added to the output of the decoding seen above. It is this sum that will be our super resolved frame directly SR_i. This sum can in fact be interpreted as a skip connection. Indeed, LR_i bypass the whole encoding and decoding part. Except that the interpolation is necessary to make sure that the 2 signals have the same size.

4.1.3 Training

This section develops some interesting aspects of the training of the BasicVSR model. Further details, such as the hyperparameters, can be found in Appendix 8.1.

Loss Function

The model training is guided by a traditional pixel loss. It measures the difference at pixel level between two images. The most common ones are mainly $\mathcal{L}1$ loss (i.e. mean absolute error) and $\mathcal{L}2$ loss (i.e. mean square error). Compared to the $\mathcal{L}1$ loss, the $\mathcal{L}2$ loss penalizes larger errors but is more tolerant to smaller errors. This often results in an overly smoothed result. In practice, $\mathcal{L}1$ loss has better performance and convergence than $\mathcal{L}2$ loss, but remember that since the definition of PSNR is strongly correlated to the difference between pixels, it is obvious that minimizing the pixel loss directly maximizes the PSNR. Therefore, the pixel loss progressively became the most used loss function.

More precisely, what is used here is a variant of the $\mathcal{L}1$ loss, namely Charbonnier loss [11] given by :

$$\mathcal{L}_{out} = \sum_{i=0}^{L} \mathcal{L}_{out,i} \quad \text{where} \quad \mathcal{L}_{out,i} = \frac{1}{N} \sum_{j=0}^{N} \sqrt{(SR_{i,j} - z_{i,j})^2 + \epsilon^2} \tag{4.8}$$

where $\mathcal{L}_{out,i}$ is the output loss associated to the i^{th} input frame. The indexes i and j denotes the j^{th} pixel of the i^{th} frame, respectively. N denotes the number of pixels and L the number of frames in the sequence. From then on, $SR_{i,j}$ is the j^{th} pixel of the i^{th} super-resolved frame, as well as, $z_{i,j}$ is the j^{th} pixel of the i^{th} high resolution frame.

This loss function is preferred to a more traditional loss because it manages outliers better and improves performances [19]. In fact, the Charbonnier loss is based on the $\mathcal{L}1$ norm where ϵ, a little constant, has been added for numerical stability. Chan et al. have fixed ϵ to 1×10^{-8}.

Parameters

As explained in the introduction, BasicVSR is a relatively lightweight model. Indeed, it is not composed of numerous trainable parameters. As shown in Figure 4.12, BasicVSR outperforms other super-resolution models in terms of quality proportionally to the number of parameters and consequently to the computation time. This is thanks to the fact that the model is recurrent. As shown in Figure 4.1, the model is in fact made up of a single level which includes a backward propagation module, a forward propagation module and an upsample module. The 2 propagation modules are architecturally identical but are 2 different entities and therefore have different parameters which will each be trained. In practice, no matter how many images are present in the input sequence, this single level will be repeated (with the same parameters) for each image. Note that this means that the number of parameters to be trained is not proportional to the number of images entered, but that having a long sequence still requires more memory resources since every image have to be loaded.

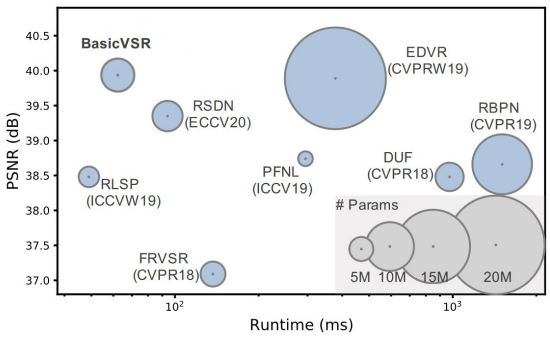


Figure 4.12: Speed and performance comparison of state-of-the-art methods performed on UDM10 dataset from Chan et al. [10]

To let these parameters converge, the learning is based on Adam's optimizer but with the particularity to be encapsulated in a Cosine Annealing Scheme. The Cosine Annealing Scheme is a modification of the learning rate behaviour. Its objective is dual. It combines periods with a high learning rates and periods with low rates. The period with the high learning rate prevent the learner from getting stuck in a local cost minima. The periods with the low learning rate allow the scheduler to converge to a near-optimal point. In practice, the learning starts with a relatively high initial learning rate η_{max}, but quickly decrease following a cosine form to converge to a minimum value η_{min}. This annealing is given by :

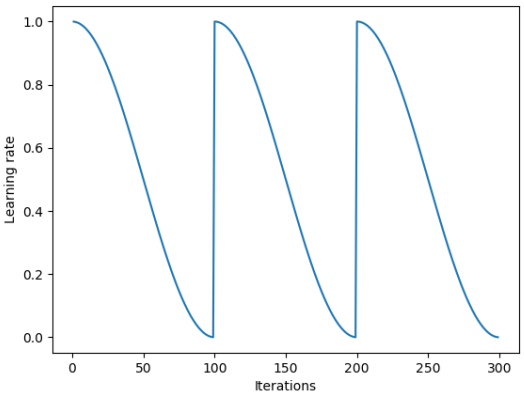


Figure 4.13: Plot of the cosine rate annealing over each epoch by Mishra et al. [22]

$$\eta_t = \eta_{min} + \frac{1}{2}(\eta_{max} - \eta_{min})(1 + cos(\frac{T_{cur}}{T_t}\pi)) \tag{4.9}$$

where T_{cur} is the number of steps since the last restart and T_t is the number of steps between two restarts. So, when $T_{cur} = T_t$, $\eta_t = \eta_{min}$ to mark the end of the descent before restarting by setting T_{cur} to 0 where η_t will be back to η_{max}. This can be seen as a restart of the learning process, not from scratch but from already relevant weights (called a "hot restart").

Datasets

The authors of BasicVSR trained the model with 2 large datasets, namely, REDS [25] and Vimeo-90K [34] [4]. To train the model, these datasets must be composed of couples of low and high resolution videos. To achieve this, the images are downsampled by a factor of 4 using 2 degradations, first a bicubic followed by a blur downsampling. As it will be explained later in section 4.2, these two degradation seems to be slightly too simple compared to real images shot in low quality.

[4]REDS is composed of 300 videos where the high quality frames are in HD (1280×720) images and Vimeo-90K-T is composed of 89,800 video clips of various resolutions downloaded from `vimeo.com`

4.1.4 Results

As in any project, several constraints had to be solved, such as GPU limitations but also some errors directly linked to the model. Moreover, once these problems were overcome, the results obtained were still not necessarily up to our expectations.

GPU Limitations

As using this model on a classical home computer would be very slow, the services of the CÉCI organization were made available to us. It groups together various computing clusters, such as the Dragon2 cluster that was used to train and run the model. Although this cluster has 2 NVIDIA Tesla V100 GPUs with 16 GB each, it was limiting our work. Indeed, it was not possible to run the model on the original dimensions of the images from Agile Birds, which is a 4K resolution (3840×2160) because they saturated the GPU memory.

To counter this problem, the first thing that was done was to work with images whose spatial dimensions were reduced by a factor of 4. In our case, the resulting images had spatial dimensions of 960×540. Also, the length of the entry sequence was limited to 20 frames to obtain the first results that are visible in Figure 4.20.

However, it is not always practical to manually limit the length of a sequence. To deal with this limitation, it is possible to treat the complete sequence as a succession of sub-sequences. In practice, the complete sequence is loaded into main memory but only sub-sequences are sent one by one to the GPU to be super-resolved separately. The results of each sub-sequence are stored in main memory too and then concatenated to produce the final result.

More formally, let's define a variable l that denotes the maximum length of the sub-sequences, so that the execution of the model (in this case BasicVSR, noted $BVSR$) on that sub-sequence does not overload the GPUs. The following equation indicates which frames will make up the m^{th} sub-sequence.

$$SR_0^m, SR_1^m, \ldots, SR_{l-1}^m = BVSR(LR_{m \times l}, LR_{m \times l+1}, \ldots, LR_{m \times l+l-1}) \quad \forall m \in [0, L//l - 1] \tag{4.10}$$

Therefore, notice that when the number of frames is not a multiple of the size of the sub-sequences, the last one may be shorter than l. Now, the following equation indicates which frames will make up the last sub-sequence.

$$SR_0^m, SR_1^m, \ldots, SR_{(L\%l)-1}^m = BVSR(LR_{m\times l}, LR_{m\times l+1}, \ldots, LR_{m\times l+(L\%l)-1}) \quad m = L//l \tag{4.11}$$

Where SR_i^m is the i^{th} output frame of the model applied on the m^{th} sub-sequence. LR_i denotes each frame of the input video. The length L is the number of frames that compose the input video. $L//l$ denotes the integer division of L by l and $L\%l$ is the modulo operation that associates the rest of the Euclidean division of L by l. The output is then the concatenation of all these super-resolved frames :

$$SR_0^0, \ldots, SR_{l-1}^0, SR_0^1, \ldots, SR_{(L\%l)-1}^{L//l}$$

Note that the last sub-sequence produced may be of lower quality than the others, as the model will have access to fewer images and therefore the propagation will be impoverished. Moreover, the use of shorter sub-sequences decreases the advantage that propagation could bring, as it is then reduced to the size of the sub-sequences. In addition, there will be several frames at the limits of the sub-sequences, i.e. the first and last frames of a sub-sequence, only benefit from a unidirectional propagation as the propagation is cut off between the sub-sequences. However, please note already that in the continuation of this work, we will see that it is not always the case.

Propagation Error

The power of BasicVSR comes from the use of long-term information through propagation. In other words, it uses the information from all frames to super-resolve each of them. However, with that kind of architecture, the model could accumulate errors during propagation and generate artefacts [10]. This is what happened when we tried to run the model on a relatively long sequence (369 frames) without limiting the size of the sub-sequence ($l = L$). In order to respect the GPU limitation, it was necessary to downsample this sequence to the spatial dimension 174×98.

Figure 4.14 shows 3 frames of this run whose output has degenerated. In more detail, the beginning of the output sequence is normal, as it can be seen in the first frame on the left. In fact, the explosion has not yet appeared, but we can already see a certain instability in the image texture. It is quite quickly, around the 10th frame, that the explosion appears exactly at the place of the fire. The explosion takes the form of pixels with very high intensities and spreads over the whole image in a few frames to give way to a totally black image which will persist until the end of the sequence.

At first sight, this effect seems to come from an error that explodes and whose effect is amplified by its temporal propagation. One can imagine that the image being of very low quality, it is potentially a source of large impurities which will be misinterpreted and become artefacts. Add to this the fact that the sequence is very long and not limited by the sub-sequences, this error can propagate over the entire length of the sequence and thus accumulate.

Figure 4.14: Occurrence of an instability in the output of BasicVSR with as input a video composed of 369 downsampled[5] frames to a size of 174×98. The upper line is composed of some input frames ($1^{th}, 19^{th}, 61^{th}$) that have been stretched to have the same visual dimensions as the super-resolved images below. Zoom-in for a better view

To verify the above hypothesis, the first thing that was done was to determine the cause of the black screen that occurs after the explosion. Contrary to the classical encoding of a black pixel which is a null value on the 3 channels (red, green, blue), here, they are the consequence of `NaN` values. Indeed, Figure 4.15 shows that the sum of the pixel intensities grow exponentially from a few frames before the one at index 12. These values increase until they reach values higher than what can be encoded by integers. `Numpy` then replaces the value with `inf`. The problem arises when mathematical operations are performed with these `inf`. Since these are not defined, the results of the operations are `NaN`.

After further investigation, as shown in Figure 4.16, h_i^f, the output of the forward module follows the same trend but starts a few frames before. This lag is probably due to the fact that the explosion is initially dampened by the rest of the network. Indeed, the backward propagation does not suffer from this growth and the work of the upsampling module will dilute the effect that the information propagated

[5]The downsampling was done using the `cv2.rezise` method with `cv2.INTER_LINEAR` parameter

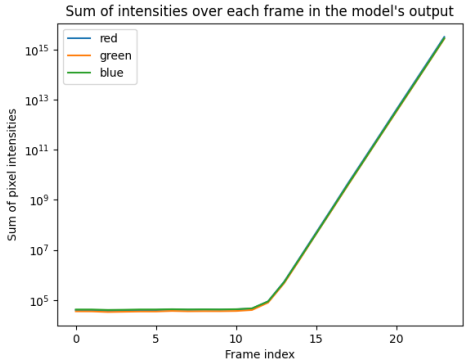


Figure 4.15: Sum of the values for each channel in the degenerated output. Only the first 24 frames are shown on this graph to provide a clearer illustration.

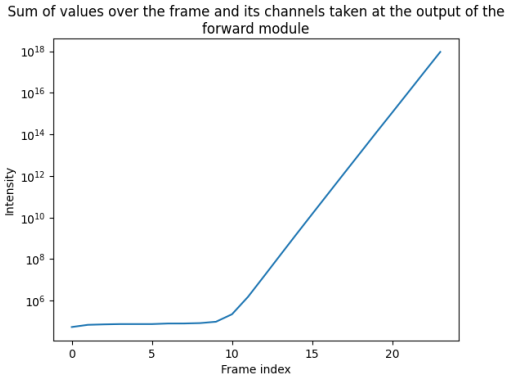


Figure 4.16: Sum of the values of each channel taken at the output of the forward module during the explosion. Only the first 24 frames are shown on this graph to provide a clearer illustration.

by the forward branch may have. Remember that the upsampling module takes advantage of the information from the forward module h_i^f but also the backward module h_i^b and the input image LR_i.

To find out which part of the network is the first to become unstable, each module of the forward propagation branch was analysed. Figure 4.17 highlights the fact that the explosion first occurred in the residual module. Indeed, at level 10 while its input (the output of the warping module w_{10}^f) is still stable, the output of the module starts to increase.

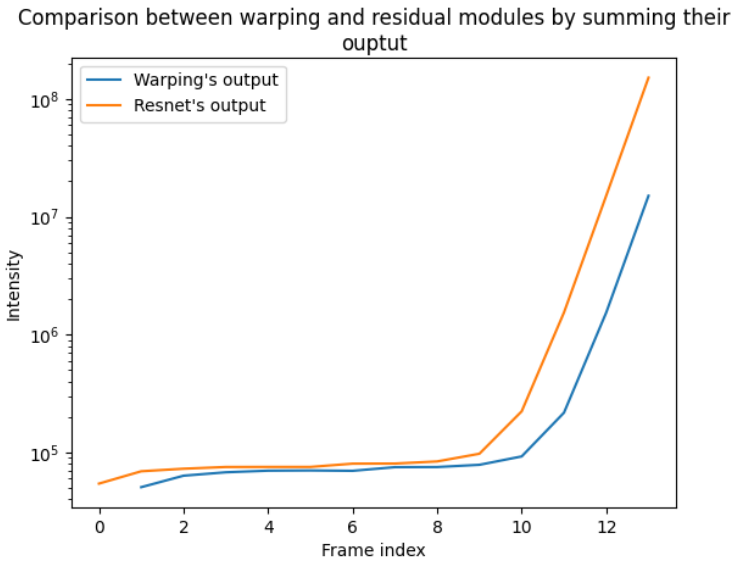
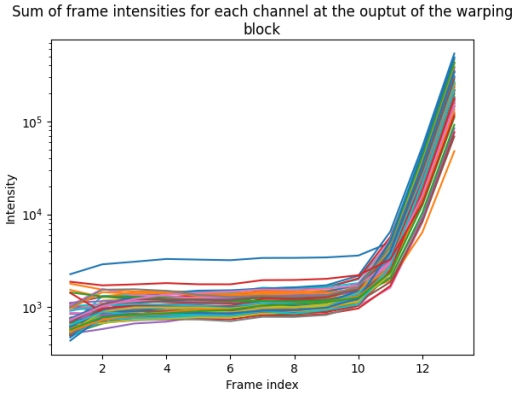
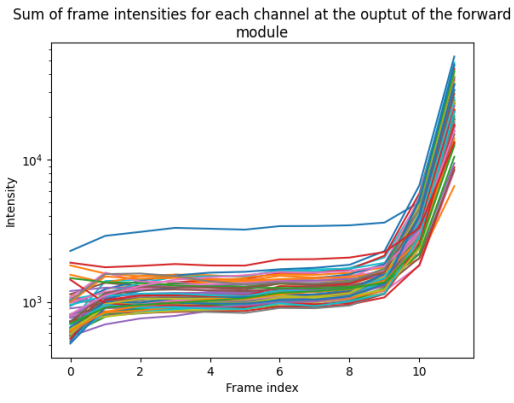


Figure 4.17: Comparison between the output of the warping and the residual module. This result has been obtained by summing all channels and values for each frame during the unstable inference shown on Figure 4.14.

Despite that, we cannot directly eliminate the hypothesis that the problem would start earlier in the model. Figure 4.17 shows the average values of all the features, so maybe one of them is still a problem. To be sure, the average values of the features have been unpacked and shown in figure 4.18.



(a) Sum of the values for each channel in the output of the forward warping module of BasicVSR.



(b) Sum of the values for each channel in the output of the forward module of BasicVSR.

Figure 4.18

After that, it appears that a feature is, indeed, taking higher values than the average. Despite more precise investigations, it is difficult to give a real meaning to this feature. That's the whole principle of machine learning, especially deep learning, we don't really know what the features are.

Beyond that, what is being analysed here is the output of the warping and therefore the application of flow to the encoding of the images. The flow could very well take completely wrong values without blowing up the channel values. The only thing that could happen is an abusive shift in values. To investigate this, the flow values were analysed and displayed over the current frame. Figure 4.19 illustrates this.

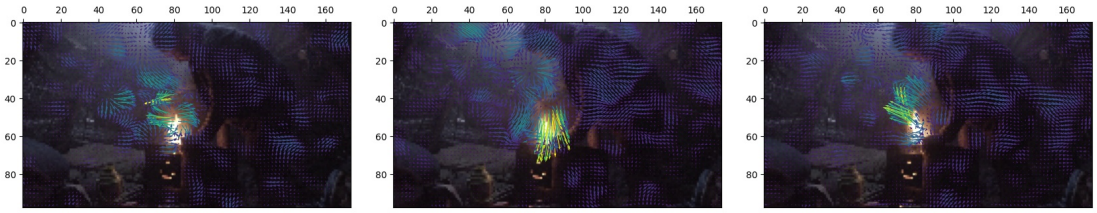


Figure 4.19: The flow of the frames at index 9, 10, and 11 in BasicVSR of the video presented in Figure 4.14. The flow is overlaid on the images and illustrated as arrows of which the size are 10× longer than the norm for illustration purposes.

Two things can be deduced from the information found on the flow and partly illustrated in these images. Firstly, the flow does indeed detect movement in areas where there is not supposed to be any (e.g. the background of the sequence). This supports the idea that the poor quality of the images involves impurities that are misinterpreted by the network and generate artefacts. Secondly, notice the difference in the values of the flow in the fire. These are relatively large and change a lot from one frame to the next. This confirms the fact that the flow is always a potential source of the problem.

To summarize, one could suppose that the source of the explosion is a combination between the presence of a high variation of pixel intensities, due to the representation of the fire, and the instability of the flow, due to the poor quality of the input. This problem could therefore be avoided by, for example, reducing the size of the sequence or splitting it into several sub-sequences reduced to a size $l < L$, to reduce the impact of propagation. Another solution would be to find a way to directly limit the impurities in the input image. This is what will be developed in section 4.2. Moreover, the situation described here will be repeated with these improvements and the results will be compared in section 4.2.4

Qualitative Results

Figure 4.20 shows the output of BasicVSR run on one of the Agile Birds sequences that we have been asked to enhance. It compares the input with the ground truth and the output of the model, respectively, on the top, in the middle and on the bottom line. The first column shows the frames of the sequence used, while the other two columns are zooms of certain areas of this image. In particular, the model was run on a sequence of 369 frames downsampled by a factor of 4 to arrive at a resolution of 960 by 540 pixels with a maximum sub-sequence length of 20. Indeed, despite the use of shorter sub-sequences, it was not possible to use the original images directly as model input without saturating the GPU memory. That said, notice that some areas of the image are of better visual quality. Take for example the bottle in the second zoom, that is sharper. So it's good to see that the task of super-resolution has been accomplished, and that in the end we have images with high resolution and better visual quality in some areas.

Figure 4.20: 150^{th} frame obtained by BasicVSR on a 369 frames long input video downsampled to 960×540 images with $l = 20$. For clarity reasons, the input image has been stretched to have the same visual dimensions as the super-resolved image. Zoom-in for a better view

Despite the results announced by the authors as being as good or even better

than the State-of-the-Art, we notice that in our case, the images obtained are not satisfactory. Indeed, the visual improvement is only present in a few small places. In fact, if you look closely, you can see an effect that is present on almost the entire image. Notice the sort of rough texture of small vertical ripples that appears in the super-resolved image. This is particularly pronounced in the darker areas, where the colours are relatively uniform.

4.1.5 Summary

The above analysis develops the design choice of BasicVSR. For propagation, the model incorporates bidirectional propagation. By having as many layers as there are images in the video sequence or sub-sequence, it emphasizes long-term and global propagation. For alignment, BasicVSR adopts a simple feature level alignment based on the flow computed by SPyNet. Finally, the upsampling has a simple architecture that is based on popular choices of features concatenations done by convolutions and PixelShuffle operations. These choices, although simple, allow BasicVSR to achieve good performances in terms of quality and efficiency in classical task of super resolution. However, the model struggles to repeat these performances on more complex images such as those we are concerned with. Fortunately, the architecture of BasicVSR is also very versatile and allows to easily integrate additional modules to cope with that kind of various constraints.

4.2 RealBasicVSR

The problem with BasicVSR is that these results are poor on real world images, at least on the real images provided by Agile Birds... As it can be seen from the BasicVSR results shown in Figure 4.20, an unwanted effect is present on almost the entire image. One hypothesis for the reason of these impurities is the exaggeration of the noise effect from which the images used suffer. More generally, the apparition of certain artefacts and degradations by the model is a known problem in super-resolution models, especially when applied to real-world images. A first solution to deal with that was to use a new version of the model adapted to real images, named RealBasicVSR proposed by Chan et al. [10]

The majority of super-resolution models are trained on datasets consisting of pairs of high and low resolution images. To obtain these datasets, the technique used is to take the original as high-resolution images, degrade them in some way and then subsample them to obtain the low-resolution images. Only, by doing so, the model learns to deal specifically with artificially imposed degradations. It would then solve a simplified task, since real videos contain a wide range of unknown degradations. Therefore, it cannot be generalized and this may lead to the appearance of artefacts. To avoid this, RealBasicVSR, degrades itself the data at training time with a random sequence of degradations. In practice, several degradations are applied in the same order for every frame, but the parameters of these degradations are determined randomly. This way, the model cannot simply learn to reverse a specific degradation.

Besides, BasicVSR has a long term propagation since it uses as many frames as the length of the sub-sequences to super resolve each frame. Although this is beneficial for restoration, it can also be one of the causes of exaggerated artefacts, due to the accumulation of errors during propagation. Therefore, RealBasicVSR is investigating some trade-offs in the batch size and the sequence size to reduce the occurrence of artefacts and training time without losing quality.

4.2.1 Global Architecture

In addition to proposing a stochastic degradation of the data, Chan et al. [10] propose a simple modification to the model architecture. As shown in figure 4.21, this new version of the model integrates an additional module before the complete initial architecture of BasicVSR. This implies that, apart from what will be explained in this section, everything that has been said about BasicVSR in the previous sections remains valid here.

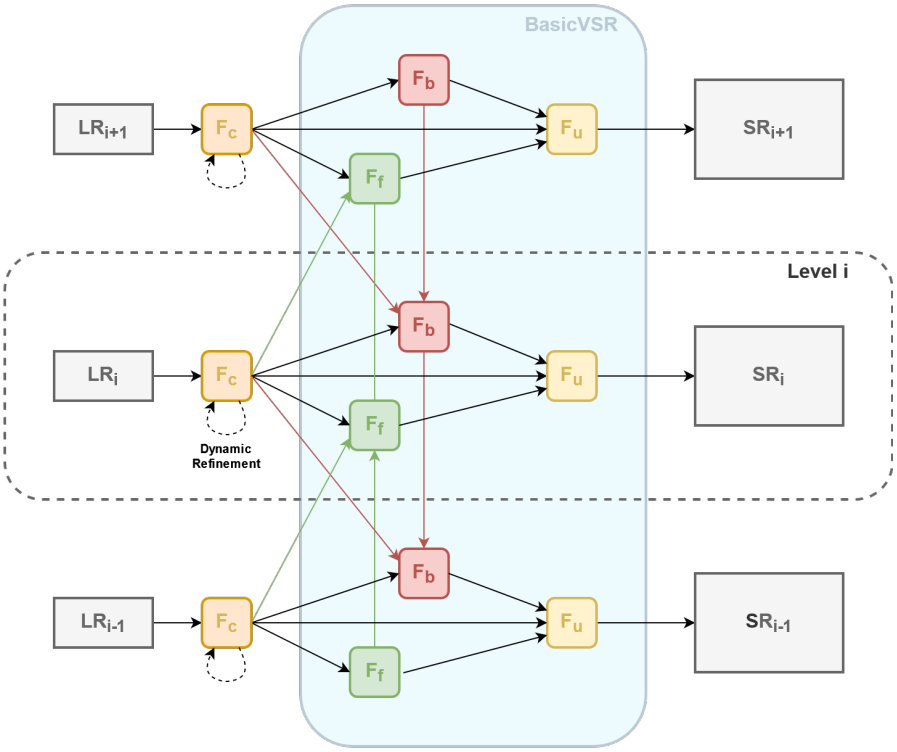


Figure 4.21: RealBasicVSR Global Architecture

This module, called the cleaning module and noted F_c, will be used to recursively smooth the input at each level. This is done to eliminate any kind of inconsistencies present in the input images due to their low quality. However, this comes at a price. Too much smoothing removes certain details, as it can be seen in Figure 4.22.

Figure 4.22: Example of the effect of the cleaning module

For example, the texture of the cushion at the bottom left of the image tends to disappear after a few passes through the cleaning module. Therefore, a dynamic refinement technique is implemented. In practice, the cleaning module is applied repeatedly to the same image. To avoid any excessive smoothing, the number of applications of that module is adjusted according to a predefined threshold detailed in the equation 4.13 . This allows a flexible compromise between smoothness and detail.

4.2.2 Detailed Architecture

Figure 4.23 shows the detailed architecture of the cleaning module. It consists of a stack of residual blocks with an input convolution which is used to obtain the internal dimensions of the residual network (from 3 to c channels). In fact, this is the same structure as the one used in the residual module of the propagation module in BasicVSR. In practice, an output convolution has been added to restore the same dimensions as the input image (from c to 3 channels). Also, note the presence of a skip connection that bypasses the entire module. In other words, the cleaning module is a network that only calculates the residual of the mapping between the input and the cleaned image.

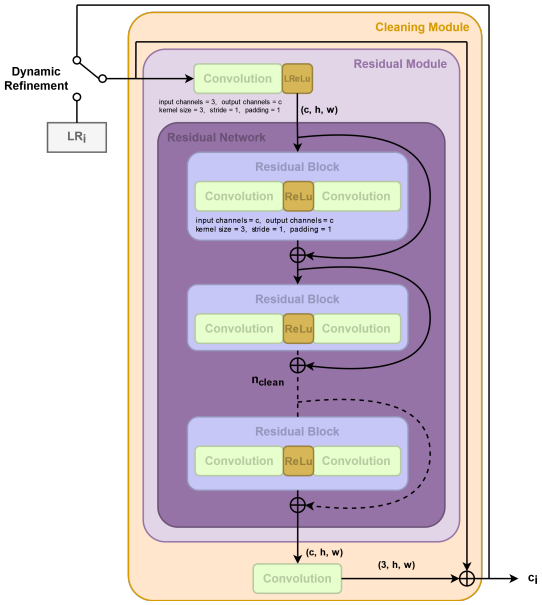


Figure 4.23: Detailed Architecture of the Cleaning Module

The execution proceeds as follows. An image is sent to the cleaning module to eliminate the degradation of the low resolution frame. Again, let LR_i be the i^{th} image of the input sequence, and F_c our cleaning module.

$$c_i = F_c(LR_i) \tag{4.12}$$

Then the stop condition is checked to see if the image should go through the cleaning module again. More rigorously, the cleaning mechanism behaves as follows:

$$\begin{cases} c_i^{p+1} = F_c(c_i^p) & \text{if } mean(|c_i^p - c_i^{p-1}|) \geq \theta \\ c_i = c_i^p & \text{otherwise} \end{cases} \tag{4.13}$$

With the p representing the p^{th} pass in the cleaning module and therefore $c_i^0 = LR_i$. θ is a pre-determined stopping criteria fixed to 1.5 by Chan et al. [10]. Once the stop condition is met, the cleaned image is passed to the BasicVSR network for the super-resolution task:

$$SR_i = BasicVSR(c_i) \tag{4.14}$$

Finally, it is important to note that although this module is quite simple, it could not be conceptualized in an arbitrary way. RealBasicVSR arrived at this module as a result of an ablation study that compared different architectures, but also different loss functions. What is used here is the result of choices made to balance the presence of detail and the occurrence of artefacts and, of course, the efficiency of the network. Therefore, the number of residual blocks in the residual module in BasicVSR has been reduced to maintain a comparable complexity while reaching better scores.

4.2.3 Training

Except for the cleaning module, the architecture of RealBasicVSR is identical to BasicVSR. The difference in performance between the two models is however quite significant. In fact, other things can improve a model, as it is the case here thanks to the training which has been improved in different ways which are explained below.

Loss Functions

To achieve the desired result, the model is trained in two steps. First, the complete model (BasicVSR + Cleaning Module) is trained following 2 losses.

$$\mathcal{L}_{1^{st}} = \mathcal{L}_{out} + \mathcal{L}_{clean} \tag{4.15}$$

The first one, $\mathcal{L}_{out}$, is the same as the one used in BasicVSR, i.e. the output fidelity loss and is given by :

$$\mathcal{L}_{out} = \sum_{i=0}^{L} \mathcal{L}_{out,i} \quad \text{where} \quad \mathcal{L}_{out,i} = \frac{1}{N} \sum_{j=0}^{N} \sqrt{(SR_{i,j} - z_{i,j})^2 + \epsilon^2} \tag{4.16}$$

The second one concerns the cleaning aspect of the low resolution image.

$$\mathcal{L}_{clean} = \sum_{i=0}^{L} \mathcal{L}_{clean,i} \quad \text{where} \quad \mathcal{L}_{clean,i} = \frac{1}{N} \sum_{j=0}^{N} \sqrt{(c_{i,j} - d(z_{i,j}))^2 + \epsilon^2} \tag{4.17}$$

Here, the Charbonnier loss is used to compare the cleaned image c_i with the ground truth high resolution image z_i, which is itself degraded, with d a downsampling operator [6]. From then on, $\mathcal{L}_{clean,i}$ is the cleaning loss associated to the i^{th} input frame. $c_{i,j}$ denotes the j^{th} pixel of the i^{th} cleaned frame, as well as, $z_{i,j}$ is the j^{th}

[6] corresponding to the `area` mode of the `interpolate` PyTorch function

pixel of the i^{th} high resolution frame. N denotes the number of pixels and L the number of frames in the sequence.

The second step of the training consists in refining the visual quality using the perceptual loss $\mathcal{L}_{per}$ and the adversarial loss $\mathcal{L}_{adv}$. At this stage, the weights of the cleaning module are kept fixed and the rest of the model is trained further according to :

$$\mathcal{L}_{2^{nd}} = \mathcal{L}_{out} + \mathcal{L}_{clean} + \lambda_{per}\mathcal{L}_{per} + \lambda_{adv}\mathcal{L}_{adv} \tag{4.18}$$

Chan et al. have found $\lambda_{per} = 1$ and $\lambda_{adv} = 5 \times 10^{-2}$ [10].

These two new terms are used to emphasize the visual aspect rather than a pixel-by-pixel accuracy. In other words, the model will not be penalized if 2 pixels do not match between the ground-truth and the model output. The aim is to have pixels that are relevant but not necessarily identical to the ground-truths.

The perceptual loss that is represented by the 3rd term can be interpreted as a pixelwise loss but that is calculated on the features of an image. In particular, a dedicated pre-trained network is used, the VGG-19, called the Loss Network. It is a Convolutional Neural Network model proposed by Simonyan et al. [30] that proved to be a significant milestone. The main contribution is an in-depth evaluation of networks of increasing depth using an architecture with very small (3x3) convolution filters and stride (1 pixel).

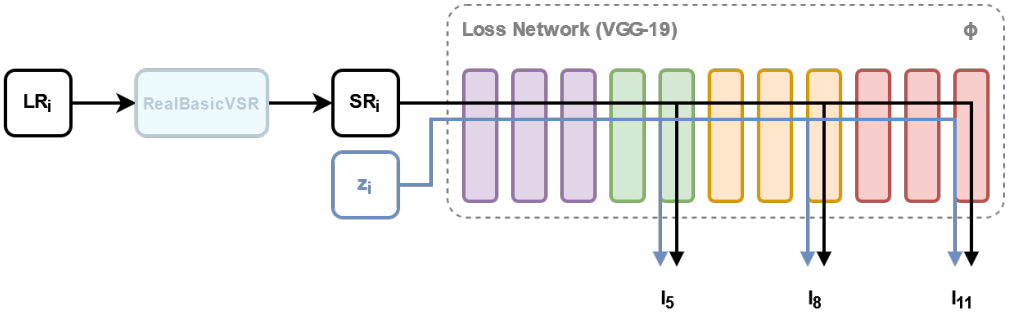


Figure 4.24: Example of the Loss Network and the layers involved in the perceptual loss calculation

As shown in Figure 4.24, it is composed of several layers. The prediction of our model SR_i as well as the corresponding ground-truth z_i will be sent to this model to be encoded. The encoding of these two images will be compared together at a set K of different layers of the Loss Network ϕ according to the following formula :

$$\mathcal{L}_{per} = \sum_{i=0}^{L} \mathcal{L}_{per,i} \quad \text{where} \quad \mathcal{L}_{per,i} = \sum_{k \in K} \lambda_k l_k(SR_i, z_i) \tag{4.19}$$

where

$$l_k(SR_i, z_i) = \frac{1}{N_k} ||\phi_k(SR_i) - \phi_k(z_i)||_2^2 \tag{4.20}$$

with $\phi_k(x)$ the activation of the k^{th} layer of the Loss Network and N_k the number of elements of the tensor of the encoding at the k^{th} layer [7]. The K layers are chosen arbitrarily as well as associated to the λ_k.

Then, to understand the last term, we have to remember what a Generative Adversarial Network is (GAN). As explained in the State-of-the-Art, a GAN is an architecture composed of 2 parts, namely, the discriminator and the generator. The aim is to alternatively train the generator to generate images and the discriminator to determine whether an image comes from the real image dataset or whether it was generated by the generator. Once the entire architecture has been trained, usually only the generator is used since it is the useful part for image generation. But, in the case of the $\mathcal{L}_{adv}$, it is the discriminator that will be used. This loss is based on the Real-ESRGAN discriminator [31] which is the reference model in the field of Single Image Super Resolution. In practice, this discriminator has been trained to determine whether an image has been super-resolved. Therefore, by giving it our super-resolved images as input, our model can play the role of the generator, and thus be trained.

More precisely, in the case of one training step, each super resolved images from the batch will be sent to the discriminator for evaluation. The discriminator will return a continuous value between 0 and 1. The closer the value is to 0, the more the image will be considered as false. Conversely, the closer the value is to 1, the more the image will be considered as a true image coming from a data set. Once this is done, these predicted labels $\hat{y}$ will be compared with the real labels y (them, being binary, equal to 0 or 1) with a loss called Binary Cross Entropy (BCE). It is this value that the term $\mathcal{L}_{adv}$ term will take.

$$BCE = -\frac{1}{B} \sum_{i=0}^{B} [y_i \cdot log(\hat{y}_i) + (1 - y_i) \cdot log(1 - \hat{y}_i)] \tag{4.21}$$

where y_i is the true label of the i^{th} image, i.e. if the images comes from a dataset or if it is generated. $\hat{y}_i$ is the predicted label of the i^{th} image. B refers to the batch size.

[7] N_k is obtained by multiplying each dimensions

In this case, since all the images are generated, their true label y are all "fake". By defining the fake label as being equal to 0 and the true labels as being equal to 1, the equation simplifies to :

$$BCE = -\frac{1}{B}\sum_{i=0}^{B} log(1 - \hat{y}_i) \tag{4.22}$$

However, you may have noticed that optimizing this objective for the generator does not work. Indeed, imagine the case where all super resolved images are excellent and trap the discriminator which evaluates them as real images($\hat{y} \approx 1$). Then, the value inside the logarithm will be small and give a strongly negative value. With the negative sign in front of it, we finally get a large BCE value.
Conversely, for super resolved images of lower quality that the discriminator evaluates as false ($\hat{y} \approx 0$), the value inside the logarithm will become almost equal to 1 and $log(1) = 0$. No matter the negative sign, we finally get a BCE value near to 0.
In short, the gradient is very small when the example is probably false, whereas the gradient is very large when the sample is already very good...

The technique used here to deal with this, is to change the assignment of the real or fake values. In this case, rather than defining the true label y as $fake = 0$ and $real = 1$, the values are swapped to $fake = 1$ and $real = 0$. Note that this is only done for true labels y. The discriminator will always give a value $\hat{y}$ close to 0 to evaluate a sample as unrealistic and conversely.

In the same example as above, the true labels y_i are now all equal to 1 which simplifies the equation as follows :

$$BCE = -\frac{1}{B}\sum_{i=0}^{B} log(\hat{y}_i) \tag{4.23}$$

This one, now, gives a large positive value when the images are evaluated as false ($\hat{y} \approx 0$) and a small value when the images are of good quality ($\hat{y} \approx 1$).

Trade-offs

Models that are applied to real images must be able to handle a wide range of degradations. For this, their training is more complex and requires longer batch sizes to maintain a stable gradient. Therefore, since the computational budget is increased, two aspects are analysed, namely, the tradeoff of speed/performance and the tradeoff of batch size/sequence length.

According to Chan et al. [10] VSR models are usually trained on datasets generated from high resolution images and their corresponding degraded image using a wide range of degradations to improve generalization. Furthermore, bigger batch sizes are used for two reasons. First, it enables the network to perceive more degradation and scene content in each iteration. Then, it also increases the chances to keep the gradients stable. However, we have seen in previous sections that computational resources are limited. To train on bigger batch size while limiting the amount of data used in each iteration, one could use shorter video sequence as input. However, this will limit the amount of information given to the model for one iteration and therefore degrade the quality of the training. In fact, the creators of RealBasicVSR discovered that it was better to prioritize the sequence length rather of the batch size. Indeed, networks trained on longer sequences could take advantage of the information through propagation. [10].

Since loading *batch size* $\times$ *sequence length* frames from disk at each training iteration takes time, it introduces an I/O bottleneck[8]. A solution had to be found to limit exchanges with the disk. What Chan et al. did was to trick the size of the sequences. Instead of taking the L images that make up a sequence, they loaded the first $L/2$ images. This half sequence is then concatenated with its temporal symmetry to obtain a sequence of the same size, but whose content is in fact only the half. Doing so speeds up the training by almost 40% compared to classical training procedures as it requires fetching fewer frames from the disk.

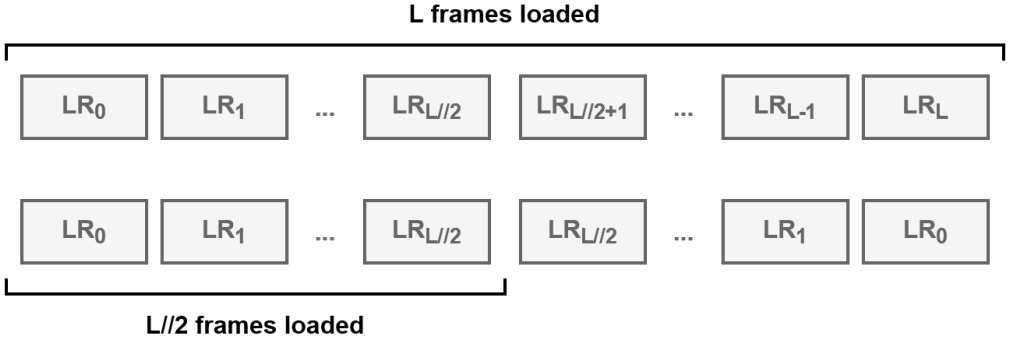


Figure 4.25: Illustration of the loading of $L//2$ frames to concatenated them with its temporal symmetry to obtain a sequence of the size L

Unfortunately, a disadvantage of this technique is that the model encounter less

[8]When a computer do not have fast enough input/output performances such that a program spend a non-negligible part of it's time waiting for input/output operations to complete, the program is said to have a I/O bottleneck

scene content, as every texture in the first half of the constructed sequences are also present in the second half. Also, the model can potentially make use of the shortcut that the sequences are temporally symmetric.

In order to partly counteract this problem, a stochastic degradation is applied to each frames of the constructed sequence. More specifically, the images are degraded following known processes but with unknown parameters. Typically, during training, the data is passed through a pipeline consisting of, among other things, degradations, compression, etc... The order in which the images pass through these transformations is known, except that the characteristics that define them are random.

Specifically, the parameters p of the degradations are defined in a random walk manner. That is, if p_i corresponds to the parameters used to degrade the i^{th} frame, the $(i+1)^{th}$ frame will be degraded with the following parameters :

$$p_{i+1} = p_i + \Delta p_{i+1}$$

Where Δp_{i+1} refers to the randomly chosen differences between the parameters from the i^{th} frame to the $(i+1)^{th}$ one.

As shown in Figure 4.26, the conventional method, shown on the left, takes all low-resolution images and degrades them in the same way. On the right, the stochastic degradation scheme can be seen by the varying p_i degradations and the fact that only 2 images are loaded for a sequence length of 4.

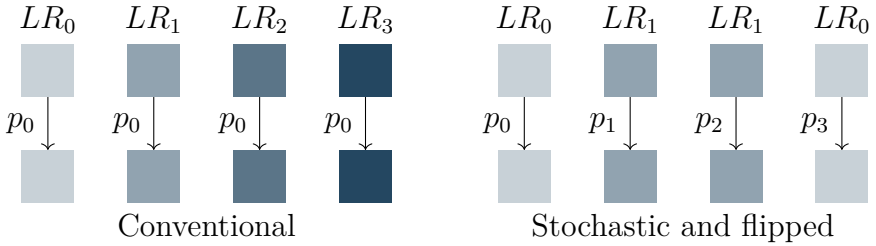


Figure 4.26: Illustration of the scheme used to preprocess RealBasicVSR's input frames. Each square represents a frame associated with a particular colour.

Datasets

This time, the model is trained on a dataset called REDS [25]. As mentioned before, this dataset is made of pairs of high resolution and low resolution images. It contains 300 video sequences of 100 frames each. The high resolution images are

1280×720 images, while the low resolution ones are 320×180 images. Those low resolution images were created by combining several degradations (including the downsampling one). However, RealBasicVSR only uses the high resolution ones and applies its degradation scheme to it as described above.

4.2.4 Results

RealBasicVSR is intended to be a more real image friendly model. Moreover, considering the disappointing results obtained by BasicVSR on our images, let's see if RealBasicVSR is really more efficient. Without forgetting that 2 aspects are taken into account: the calculation complexity and the visual quality.

Propagation Error

Remember that one of the reasons for choosing to continue the work with RealBasicVSR was the explosion problem that occurred on a long sequence of very low quality. On this one, RealBasicVSR gives much better than the those that were analysed in Section 4.1.4. Indeed, Figure 4.27 shows a much better output on the same input video.

Figure 4.27: Output of RealBasicVSR applied to the video that made BasicVSR unstable. The upper line is composed of some input frames (1st, 19th, 61th) that have been stretched to have the same visual dimensions as the super-resolved images below it. Zoom-in for a better view

Where BasicVSR was unstable, RealBasicVSR is much more stable. Indeed, its output and the one of the residual module remains in the same order of magnitude as shown in Figure 4.28. Note that the illustration of the flow values displayed over the current frame in RealBasicVSR is shown in Appendix 8.2.2.

(a) Output of RealBasicVSR

(b) Output of the forward module

Figure 4.28: Illustration of the output of the model and the output of the forward module during the inference of the video that makes BasicVSR unstable, i.e. the one presented on Figure 4.14. This result has been obtained by summing all channels and values for each frame

Based on the assumptions that were made in section 4.1.4, the stability of RealBasicVSR can be attributed to the following. Firstly, as the images are smoothed, there is less degradation and therefore the flow is not excessively computed in areas of the image without movement. Furthermore, as the model is trained with more complex degradations, it is certainly more robust to the strong degradations present here. Therefore, RealBasicVSR does not seem to suffer from the instabilities that its predecessor suffered from.

Qualitative Results

Figure 4.29 shows the result obtained by RealBasicVSR on the same sequence as the one used for BasicVSR, in Section 4.1.4. It is very satisfactory for several reasons. Indeed, we can see that the ripple effect has completely disappeared, as shown in zoom number 1. Furthermore, zoom number 2 shows how the image has an even better visual quality than the BasicVSR output. Notice for example the light reflection on the gourd, which is really sharp.

Figure 4.29: Comparison between the result obtained by RealBasicVSR and BasicVSR on a 369 frames long input video downsampled to 960×540 images with $l = 20$. The presented image is the 150^{th} frame of the sequence. For clarity reasons, the input image has been stretched to have the same visual dimensions as the super-resolved image. Zoom-in for a better view

However, there is one last slightly undesirable effect, which is the lack of detail in some places and the impression of being close to something that has been painted. Two elements can be at the base of this. The first is probably the cleaning module, which removes slightly too much detail to avoid any artefacts. The second is the fact that increasing the number of pixels requires "inventing" new ones. It is very complicated for a model to show very precise details through these new pixels. The model will tend to generate smooth surfaces rather than fine areas.

4.2.5 Summary

RealBasicVSR addresses some limitations in BasicVSR, especially around aspects specific to the application of super resolution video to real images. Indeed, Chan et al. note the presence of gaps in the area of degradations and the increase in computational costs. To this, the authors propose compromises and practical solutions, such as the cleaning module and the stochastic degradation scheme. Although they are simple, these improvements allow reaching a better efficiency, both in terms of computation time and final quality.

Chapter 5

Contribution

The world of image processing and computer vision is one domain in which deep learning has achieved astonishing results. The research in that field is still very active today. Many new techniques are being discovered that, every time, push the limits of previous work. This did not necessarily help to start this work. Foremost, the literature is huge, and many tracks have already been studied. Secondly, throughout our research, new models, each more interesting than the last, were published, which tended to rethink the work done every time. It is for these reasons that it was chosen to work with BasicVSR and its adaptation to real images, RealBasicVSR. These models do not seek to be at the edge of the technology. They want to rethink what should be considered as the basic elements of video super resolution in terms of result/calculation ratio.

5.1 Motivations

In the same vein, obtaining a simple and efficient model has been the basis of the motivations behind our contributions.However, there are two other important aspects that have also influenced our choices.
Firstly, we must not forget that the problem that is trying to be solved here is not identical in every aspect to a classic super-resolution problem. Having access to high resolution images, whether they are to be enhanced or not, allows greater freedom in how the data is used.
Secondly, it was decided not to modify the BasicVSR model for a simple reason, to keep its parameters. Indeed, these weights were obtained by the authors, who had access to resources far superior to ours. BasicVSR and RealBasicVSR have been trained on 450K iterations, which is impossible to do again in this thesis. By keeping the initial architecture of BasicVSR, and thus the trained weights, our

contribution could reach similar performances than the initial model. This allows a comparison of the models to be relevant.

5.2 Implementation

Following these 3 major reasons, a point that seemed interesting to review was the cleaning module as a whole. Indeed, although it makes RealBasicVSR more powerful on real images, it makes it less efficient. In terms of the number of convolutions, this one is relatively large. Indeed, it is composed of a residual network as deep as the one of the residual module. That is to say that the cleaning module alone represents more than a third of the convolutions of the whole RealBasicVSR model. Therefore, a first modification that could be applied to the model is the simplification of the cleaning module.

To address this, the architecture that has been chosen is a U-net, the details of which are given in Figure 5.1. This module was motivated by several reasons given below.

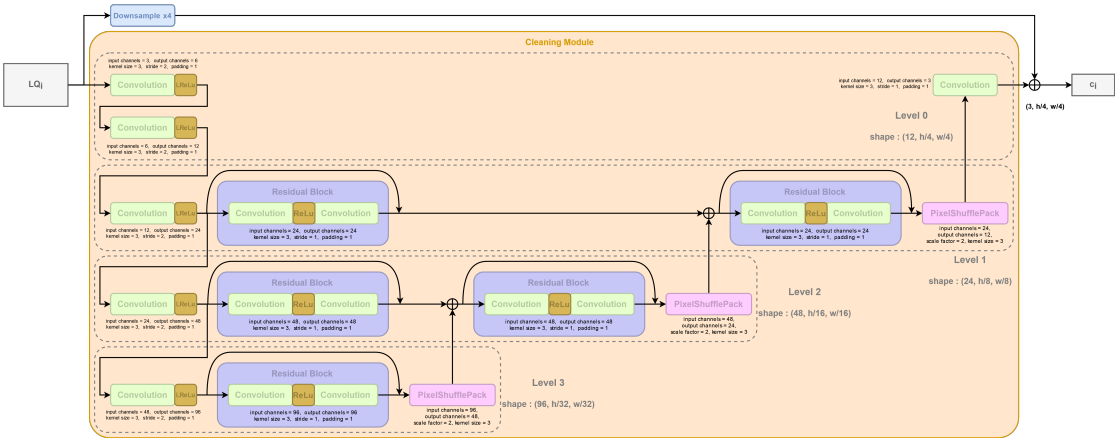


Figure 5.1: Detailed architecture of the new cleaning module

5.2.1 Coarse to Fine Spatial Pyramid Structure

Firstly, it should be remembered that what the input images suffer from is a pixelation effect, a noise effect. However, this effect is not totally uniform. In some places, usually the darker areas, it will be more pronounced and in others it will be almost absent. Figure 5.2 attempts to illustrate this with a single frame. The zoom 1 shows the pronounced presence of the undesirable effect on a face in the low-light background, while the zoom 2 shows a piece of another person's jacket with a well-detailed texture that is sharply represented.

Figure 5.2: Illustration of the non-uniformity of the pixelation effect

In response to this, a coarse to fine spatial pyramid structure is often used, such as the U-net. This involves building the model by layering several levels. Each level will work on a more downsampled version of the image. In this way, each level has to solve a simpler task, each of which concerns a different spatial scale. For example, the lowest level will receive the smallest image and can only be concerned with a coarse, global correction. In figure 5.1, the lowest level works on a 32 times downsampled version of the input images. On the other hand, the highest level, will receive the most detailed image and will have to solve a problem concerning only the finest details. In our case, the highest level receives as input a version that is already reduced by a factor of 4 so that its output has dimensions $(3, h/4, w/4)$. The reasons behind this choice are explained in the paragraph about the raw data utilization in section 5.2.

As for the levels, each is composed of 1 or 2 residual blocks, respectively, before and after the sum operation which will be used to work on the encoding of the signal. The sum operator represents the propagation of information from the lower level. This is made possible by a PixelShufflePack (see Upsampling Module in 4.1.2) which allows the dimensions to be matched.

5.2.2 Step-wise Increasing Channels

A second point where the module can be improved is the evolution of the number of channels. In several places, Chan et al. had chosen to encode their images into c channels in a single step. In other words, in (Real)BasicVSR, a single input convolution was used to go from the dimensions of an image, i.e., 3 channels (red, green, blue), to its encoding in c channels. In the case of the residual network present in the old cleaning module, this implies that each of the convolutions that compose it has c^2 kernels[1]. In the U-net that is now used, channels have been successively increased following the decrease in spatial dimensions.
Assuming that the image contains a certain amount of information, by decreasing these spatial dimensions, the amount of information contained in these dimensions is decreased. The logic is then to increase the 3^{rd} dimension to move this information from the spatial dimensions to the channel dimension. In fact, this is nothing more than an encoding, another way of representing the information. Therefore, when in the U-net the spatial dimensions are divided by 2, it seems reasonable to increase the channels only by a factor of 2 and not directly to c. In this way, the convolutions that follow will have fewer kernels and therefore be lighter and less computationally intensive.

5.2.3 Raw Data Utilization

As explained at the beginning of Chapter 4.2.5, the aim was to directly enhance the original images by redirecting the super resolution technology. Giving the raw high-resolution images to RealBasicVSR is not the best idea since it would explode the computational complexity and thus saturate the GPUs, and it would also result in images with far too high resolution, which is of no real benefit (4k is already sufficient).

What has been decided is to use the data downsampled by a factor of 4 and thus get back the original dimensions at the output while having improved the visual quality.

As such, even if it was for computational limitations reasons, this is what was done for the BasicVSR and RealBasicVSR inferences. The results obtained with these two models were done using the Agile Birds images spatially reduced by a factor of 4 using bilinear interpolation applied independently of the model.

But now, since the cleaning module has been redesigned, it has been possible to integrate the downsampling directly into it. From then on, the downsampling

[1]the amount of kernels of a convolution is given by $c_{in} \times c_{out}$ where c_{in} is the number of channels present at its input and c_{out} the number of channels wanted at its output

operation is carried out using convolutions integrated into the U-net and therefore represents an operation that is learned by the model. This allows, in a way, to keep the information, which was lost by the downsampling, inside the model. Figure 5.1 shows that in practice. The new cleaning module then takes the raw inputs, which we will now be called LQ, that stands for *low quality* (and not, *low resolution*, LR, anymore). The U-net inside it, is preceded by 2 convolutions whose aim is to reduce the spatial dimensions by a factor of 4 and at the same time to duplicate the number of channels by 4. All this information will therefore be preserved in remaining dimensions and will propagate throughout the U-net. In fact, it is in the last convolution that there will be a real loss of information. Specifically, compared to the input LQ, the output of the cleaning module will have the same number of channels (3) but will be downsampled spatially by a factor of 4. However, the objective here is to obtain an image with smaller spatial dimensions which has been cleaned up but by using all the information present in its original version. Which is the case now, since the downsampling is learned and integrated into the module. Note that this cleaning module contains a skip connection that links the input to the output. This implies that the module calculates a residue.

Still in the fashion of working with the LQs where it is possible, the skip connection in the BasicVSR upsampling module has been changed. Originally, this was done between the end of the image reconstruction and the upsampled low resolution image LR_i to match the spatial dimensions (see 4.10). But now, as shown in figure 5.3, it has been replaced by the LQ image directly since no upsampling is needed in the skip connection anymore.

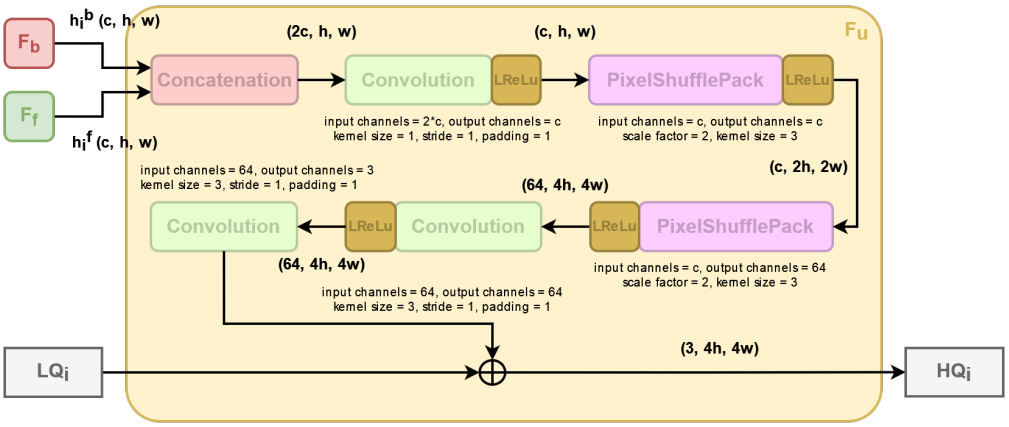


Figure 5.3: Detailed architecture of the new upsampling module of BasicVSR

This is the only change made to BasicVSR. This was left to our discretion, as the skip connection is not a learned element of BasicVSR. Obviously, each change

has its own influence on the model and the weights. As this one is minimal, we concluded that the training that would be carried out afterwards, would be sufficient to correct the potential errors induced by this modification.

Note the use of a new notation in the Figure 5.3, HQ_i which represents the output at the level i of the contributed model. HQ stands for *High Quality*, as opposed to LQ that stands for *Low Quality*.

5.3 Ablation Study

An ablation study is the term used to designate a case study between different variants of the model being investigated. In this case, it is the RealBasicVSR model plus the contributions explained and motivated above. The aim of such a study is to prove the usefulness of certain features of the model by modifying them one by one, and consequently observe (or not) a variation in the model's performance.

5.3.1 Variants

An implementation like the one proposed above is not without some subjective choices. To ensure their usefulness and effectiveness, it is necessary to test variants of the model based on these choices. Thus, the following ablation study has been carried out to clarify 3 major choices :

- The number of levels present in the cleaning module
- The number of residual blocks per level
- The presence or not of a skip connection that bypasses the entire cleaning module.

In this way, 5 models were created and are detailed in the table below:

Name	**# Levels**	**# Residual Blocs**	**Skip Connection**
`3lev_1res_wS`	3	1	Yes
`2lev_1res_wS`	2	1	Yes
`4lev_1res_wS`	4	1	Yes
`3lev_2res_wS`	3	2	Yes
`3lev_1res_woS`	3	1	No

`3lev_1res_wS` is the basic model of our contribution, as shown in Figure 5.1. As the number of levels is arbitrary, 2 variants exist, one with one extra level and one with one less level, respectively `4lev_1res_wS` and `2lev_1res_wS`.

The second point was the number of residual blocks per level. More specifically, the number of residual blocks per level before and after the sum with the rise of the lower level. To analyse the behaviour of this parameter, the basic model was modified by increasing the number of residual blocks from 1 to 2 with the `3lev_2res_wS` variant.
Finally, even if the skip connection is essentially used to manage the stability of the training, we wanted to see the effect it would have on the performance of the model. The last variant, `3lev_1res_woS`, therefore does not have one.

5.3.2 Training

In order to compare these variants with each other, they must be trained in the same way. Thus, each variant followed the same training scheme and losses as used for RealBasicVSR, as explained in the section 4.2.3. However, as we do not have the same computing power, the configurations had to be modified somewhat. For the first part of the training the number of iterations was reduced from 300,000 to 10,000 and the batch size from 16 to 3. For the second part, the number of iterations was reduced from 150,000 to 2,000 and the batch size from 8 to 1.

Note, however, a slight difference in the use of the dataset. Where RealBasicVSR used a dataset composed of pairs of LR and HR images, our contribution requires pairs of the same spatial dimensions. In order to do this, the stochastic degradation scheme that generates the LRs has been reworked to upsample them bilinearly and thus getting the same resolution as the HRs.

Once this was done, the variants were compared with each other, qualitatively but also quantitatively, by using the metric detailed in the following section 5.3.3 to determine which one is going to be considered as final.

5.3.3 Metrics

To assess the quality of each model, it is important to wisely choose the metric. A first idea, which is often used in practice, is to use the $PSNR$ which is explained in the State-of-the-Art 2.2.3. However, it compares the output images with the ground-truth using a pixelwise difference. In the situation of the images provided by Agile Birds, this is not ideal, as the images suffer from pixelation and grain. So if an output image is of excellent quality and manages to remove these unwanted effects, this difference will be penalized by the $PSNR$. This metric is therefore not relevant in our case.

Actually, this refers to the notion of perception described in the State-of-the-Art 2.2.3. In the case of our super resolution, the focus will be on the perception, since the aim here is more to enhance the perceptual quality of our images rather than producing images similar to the reference one. In fact, this is exactly the idea that motivates the PIRM-SR Challenge [2][2].

The PIRM-SR Challenge proposed a new assessment method called the Perceptual Index (PI). It is a combination of the Natural Image Quality Evaluator (NIQE) [24] and the No-Reference Quality Metric (Ma) proposed by Ma et al. [20] that were explained in the State-of-the-Art 2.2.3. This metric is calculated as follows :

$$PI = \frac{1}{2}((10 - Ma) + NIQE) \tag{5.1}$$

As a lower NIQE and a higher Ma represents a better perceptual quality, having a lower perceptual index leads to a better perceptual quality.

5.3.4 Results

Now that everything has been properly defined, this section will analyse the results, first from a subjective angle and then from an objective angle, to determine a variant that will be considered as the final model.

Removing Dynamic Refinement

It is important to mention a minor change that has been made during the analysis of the results, namely, the removal of the dynamic refinement. This means that images are forced to pass through the cleaning module only once.

[2]The PIRM-SR Challenge is one of the two most popular super-resolution challenge with NTIRE Challenge [26] according to Wang et al.[33]

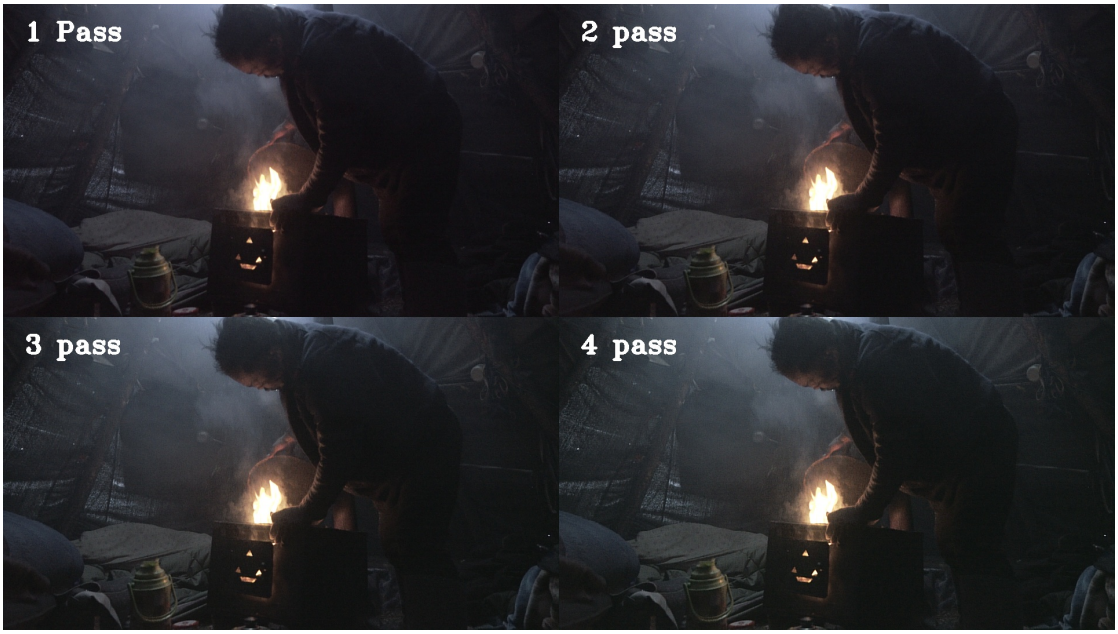


Figure 5.4: Output of the cleaning module of `2lev_1res_wS`

Indeed, after training, the behaviour of the cleaning module has been inspected for the different variants. The one of the `2lev_1res_wS` variant is visible in image 5.4. Notice that, finally, this module adds a bit of high frequencies to the image in each pass. Therefore, having more than one pass can generate too much of it and becomes counterproductive by creating artefacts... This is probably due to the fact that the aim of our contribution is actually to downsample an image that it receives as input in the best possible way so that BasicVSR can give the best results. In other words, we can assume that the cleaning module seeks to better represent the details to facilitate the task of BasicVSR.

Qualitative Results

Figure 5.5 compares, respectively, the ground-truth with the output of RealBasicVSR as well as the output of the default variant, namely `3lev_1res_wS`.

Figure 5.5: Comparison of the 19^{th} frame of the output of one of our variants with the output of RealBasicVSR and the ground-truth.

Foremost, note that the output of RealBasicVSR comes from another input. Indeed, this output is the one obtained by downsampling the ground-truth by a factor of 4. This is because, unlike our variant, it is impossible for us to obtain a result when RealBasicVSR is used on 4K images for calculation reasons. This is already a first proof of the functioning of our variant which integrates the downsampling and thus allows to have reduced images as inputs to the BasicVSR part. This is where a debate was born. How can RealBasicVSR be compared with any variant of our model as they no longer have the same function (one upsamples, the other enhances) ? One could say that you have to give the same input for these 2 models to have access to the same amount of information, but this results in totally different outcomes. RealBasicVSR would have no loss of information from its input, while our model has a loss of information at the output of the cleaning module. On the other hand, comparing (as it is done here) RealBasicVSR on a lower resolution input to match the output dimensions seems unfair since the 2 models do not start from the same information...

Then, figure 5.5 only shows the output of the `3lev_1res_wS` variant for clarification reasons, but mainly because the difference between the results obtained by the other variants are almost undetectable by eye. Section 5.3.4 will give a more quantitative answer by analysing the results obtained by each of the variants on the Pi metric explained in section 5.3.3. Appendix 8.6 still illustrates the results obtained by each of the variants.

However, we can already draw some conclusions about our contribution. Indeed, although it has not been trained a lot, the results are far beyond what was obtained by BasicVSR and its wavy outputs. In fact, it is reasonable to say that the objectives of the contribution have been achieved.
First, the fact that it was possible to run our model on ground-truths proves that it is much lighter. This is also confirmed on other downsampled images. As shown in the table 5.1, the execution time is reduced by a factor of ≈ 15 compared to RealBasicVSR run on the same input and reduced by a factor of ≈ 1.25 compared to RealBasicVSR run on an input downsampled by a factor 4.

	Total inference time	**Cleaning module**		**BasicVSR**	
		mean	std	mean	std
	On downsampled image by a factor 16 in each dimensions				
real_basicvsr	35.64	0.54	0.058	1.53	0.011
	On downsampled image by a factor 4 in each dimensions				
real_basicvsr	436.56	8.86	1.25	24.44	0.075
2lev_1res_wS	28.49	0.014	0.0013	1.512	0.018
3lev_1res_wS	27.98	0.018	0.0013	1.44	0.0086
3lev_1res_woS	28.09	0.017	0.0015	1.45	0.015
3lev_2res_wS	28.18	0.025	0.0022	1.45	0.014
4lev_1res_wS	28.09	0.023	0.0013	1.44	0.012

Table 5.1: Comparison table of the inference times of the different models and variants measured on Dragon2. The **total inference time** denotes the mean time taken on 10 inferences of a 50 frames sequences. **Cleaning module** and **BasicVSR** times denotes the average time taken by a 5-frame subsequence to pass through the corresponding part of the network. These time are also computed over 10 inferences of a 50 frame long video. All times are expressed in seconds.

Some brief remarks on the times shown in table 5.1. Note that RealBasicVSR is indeed slower than any of our variants, no matter if they are run on the same inputs or on inputs further downsampled by a factor of 4. Then, it appears that

`2lev_1res_wS` is slower than other variants despite being the lightest. Surprisingly, this is due to the BasicVSR part rather than the cleaning module, which is indeed faster.

To come back to the structure chosen for the cleaning module, it was motivated by a coarse to fine analysis. The aim was to allow the model to process the image on several scales and thus be able to work on grains and pixelation in a non-uniform way. Compared to RealBasicVSR which had the problem of smoothing the images too much, our model generates images with more precise details. In addition, even though, to be honest, the grain has not completely disappeared, it is already greatly attenuated.

Quantitative Results

Let's see, now, how the metrics can complement the first part of the analysis with quantitative and accurate information.

Name	**MSE** ↓	**Ma** ↑	**NIQE** ↓	**PI** ↓
`3lev_1res_wS`	12.8143	6.7337	5.4268	4.34655
`4lev_1res_wS`	14.4822	6.7429	5.5842	4.42065
`2lev_1res_wS`	14.7655	6.6806	4.6895	4.00445
`3lev_2res_wS`	15.2811	6.7022	5.3566	4.3272
`3lev_1res_woS`	15.7039	6.7096	5.4308	4.3606
`RealBasicVSR`	12.6091	6.073	6.2565	5.09175
`BasicVSR`	66.9038	6.7137	5.194	4.24015

Table 5.2: Comparative table of the scores obtained by each variant as well as the 2 basic models, i.e. BasicVSR and RealBasicVSR. The scores are obtained by calculating the MSE between the obtained image and the input, along with the MA, $NIQE$ and PI. The scores are an average of the scores obtained on 10 images, 2 per sequence to be improved (1 at the beginning and 1 at the end of a sequence, chosen randomly). The best results obtained for each metric are highlighted in red.

Firstly, in overall, the results of the metrics confirm that each variant has almost identical outputs.

These results could certainly be improved by further training. Indeed, the fact that each of our models has almost identical outputs can be interpreted as evidence of the poor training. Indeed, the low number of iterations and a too small batch size can strongly influence the quality of the training.

Then it is interesting to notice that RealBasicVSR obtains the best results evaluated with the MSE but is worse than each of the variants for both the Ma and the $NIQE$ (and thus, the PI). This is quite surprising since the proportion between the first part of the training based on pixelwise losses and the second part of the training based on perceptual losses, is not the same between RealBasicVSR and our models and is contrary to this logic. That is, RealBasicVSR has twice as many iterations with pixelwise losses as with perceptual losses. Whereas our variants, although they started with partly the same weights, have 5 times more iterations with pixelwise losses than with perceptual losses. In other words, our variants should be pixelwise oriented rather than perceptual oriented. A plausible justification would simply be to say that RealBasicVSR has a cleaning module that is much more trained than the cleaning module of our variants. Remembering that a pixelwise loss tends to favour smooth outputs, RealBasicVSR is proof of this.

It is interesting to note that BasicVSR still scores well in the perceptual metrics despite the fact that its output is far from being good and that its training is not based on perceptual metrics at all. This means that the ripple effect that the output suffers from is not taken into account by the perceptual losses. This means that this effect does not modify the "macro" elements and therefore the statistics analysed by the perceptual losses.

Finally, as far as the other variants are concerned, `2lev_1res_wS` seems to stand out slightly by obtaining the best $NIQE$ and will therefore be used as the final model. It is likely that one of the reasons is that since this variant is the lightest, there was less weight to update during training, and so it was potentially able to converge faster than the others. That said, it is beneficial to the initial idea of finding the lightest possible model.

5.4 Fine-tuning

To go a step further and hopefully get even better results, the best variant was fine-tuned with 3000 additional iterations. These were done on a new dataset composed of sequences that were provided by Agile Birds and that did not need to be reworked[3]. These images, being shot in better conditions, do not suffer from any particular effect. Note that this fine-tuning follows the same configurations as the second part of the RealBasicVSR training, and is thus guided by the perception-oriented loss functions given by the equation 4.18.

[3]This new dataset is composed of 21 videos of at least 100 frames (4 seconds), for a total of 4,253 `4K` frames

Unfortunately, as Figure 5.6 shows, the results obtained are no better and are even slightly worse, from a subjective point of view. Indeed, the resulting fine-tuned image is brighter and seems to have a bit more grain than the output of the final non-fine-tuned variant.

Figure 5.6: Comparison of the 19^{th} frame of the ground-truth, the output of our `2lev_1res_wS` variant and this variant fine-tuned

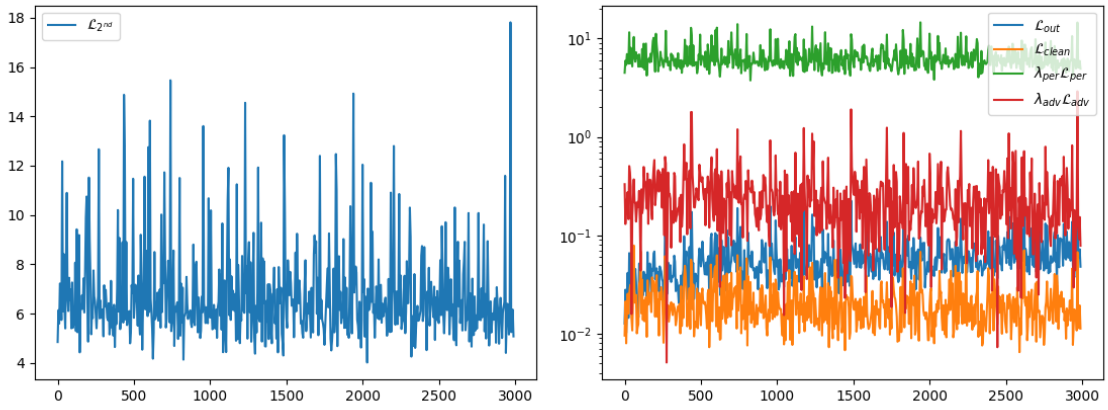
In order to be able to compare analytically, the score obtained by the metrics are presented in table 5.3. Again these scores were obtained by taking the mean of the scores obtained on 10 frames, 2 per sequence to be improved (1 at the beginning and 1 at the end of each sequence chosen randomly).

Name	MSE ↓	Ma ↑	NIQE ↓	PI ↓
`3lev_1res_wS`	12.8143	6.7337	5.4268	4.34655
`4lev_1res_wS`	14.4822	6.7429	5.5842	4.42065
`2lev_1res_wS`	14.7655	6.6806	4.6895	4.00445
`3lev_2res_wS`	15.2811	6.7022	5.3566	4.3272
`3lev_1res_woS`	15.7039	6.7096	5.4308	4.3606
`RealBasicVSR`	12.6091	6.073	6.2565	5.09175
Fine-tuned from `2lev_1res_wS`	110.9805	6.6689	5.3488	4.3400

Table 5.3: Comparative table of the scores obtained by each variant, as well as RealBasicVSR and our fine-tuned model. The scores are obtained by calculating the MSE between the obtained image and the input, along with the MA, $NIQE$ and PI. The best results obtained for each metric are highlighted in red.

Table 5.3 shows that additional iterations of the fine-tuned model did not improve the result, as model `2lev_1res_wS` scores better. In fact, although, the fine-tuned model does not perform as well as all our other variants in therm of Ma, it is better than most of them in therm of $NIQE$. Finally, the variant `2lev_1res_wS` not fine-tuned remains the best in therm of PI.

The main reason for these bad results is certainly the quality of the fine-tuning due to the very small batch size $B = 1$. Indeed, where all other training was done on existing datasets such as $REDS$ which have HD ground-truth (1280×720), this time the fine-tuning was done on $4K$ images. Note also that this one was done following the loss function $\mathcal{L}_{2^{nd}}$ 4.18 containing the terms $\mathcal{L}_{per}$ and $\mathcal{L}_{adv}$ which are more computationally intensive than the 2 other terms present in the loss function $\mathcal{L}_{1^{st}}$ 4.15. This meant that the training parameters had to be reduced to avoid memory saturation and therefore the batch size. Consequently, and as figure 5.7 shows, the training is not optimal at all and such a small batch size does not make it possible to make converge the model.



(a) $\mathcal{L}_{2^{nd}}$ (b) $\mathcal{L}_{clean}$, $\mathcal{L}_{out}$, $\lambda_{per}\mathcal{L}_{per}$, and $\lambda_{adv}\mathcal{L}_{adv}$

Figure 5.7: Evolution of the loss $\mathcal{L}_{2^{nd}}$ used for the fine-tuning of the final variant. As well as the detail of the evolution of each term that composes $\mathcal{L}_{2^{nd}}$.

5.5 Scalability

Let's not forget that the main purpose of this work is to improve the complete Agile Birds sequences. In all the previous sections, shorter sub-sequences [4] of the original videos were used for convenience. This section explains the modifications that had to be made to the model launcher in order to make inferences on long videos since the longest video to be reworked is in $4K$ and has 6708 frames.

It has already been explained in section 4.1.4 that the GPU memory saturates when inferring long or high resolution videos. To get around this limitation, the L images to be inferred were loaded into main memory, but subsequences composed of a maximum of l images were loaded into GPU memory and processed successively.

However, with $4K$ images, the main memory was not sufficient either... Indeed, loading the whole video sequence and keeping all the enhanced images before writing it back to disk caused the main memory to be saturated.

To get around this difficulty, we added a feature capable of splitting the work into packets of images in both main and GPU memory. In practice, a parameter allows to determine a size q of a subsequence to be loaded into main memory. Then an inference is executed on these q images and the result is directly written back to disk. After that, the next q images are loaded, processed, written, etc...

[4] typically 369 frames downsampled by a factor of 4 or 50 frames of original size ($4K$)

This is done by loading the model and its weights only once, because this is a time-consuming operation (about 15 seconds).

5.6 Summary

After all the analysis done on RealBasicVSR and the results of the ablation study, the `2lev_1res_wS` variant was defined as the final model presented in this work.

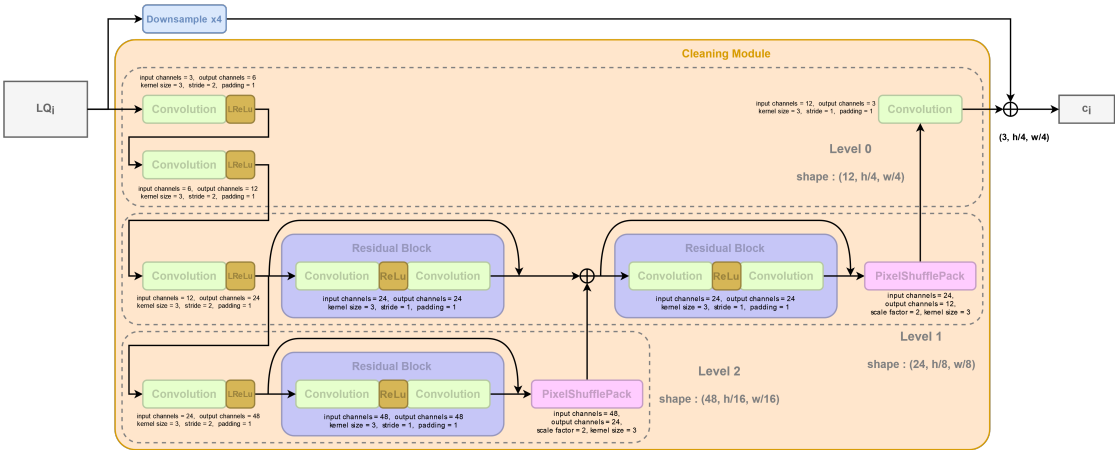


Figure 5.8: Detailed architecture of the cleaning module of the final model

Its pyramidal architecture allows the model to downsample the input image while working on its grain level and maintaining a high level of detail. Moreover, this architecture is faster than the reference mode, RealBasicVSR, by a factor of ≈ 15 at runtime on identical inputs. This is due to the fact that the majority of the model works on a reduced image size but also thanks to the number of parameters which has been reduced by 30% [5]. The results obtained by the final model on the other rushes that had to be improved are presented in appendix 8.7.

In the end, it seems that the idea of diverting the methods used in super-resolution to use them in restoration was good for 2 reasons. First, the results we obtained are satisfactory. Secondly, a recent publication that appeared in mid-April [9] confirmed the idea that was investigated in this report. Indeed, the authors of (Real)BasicVSR had the same idea of extending one of their models (BasicVSR++ [6]) to a generic framework for video restoration tasks and obtained very good results. In line with the approach followed in this paper, they reworked their

[5] RealBasicVSR : 4,854,534 trainable parameters – `2lev_1res_wS` : 3,502,462 trainable parameters

model so that inputs and outputs have the same spatial size by reducing the input resolution through strided convolutions and thus, also, maintain efficiency.

Chapter 6

Improvements

After reviewing the work presented in this thesis, it was possible to propose a non-exhaustive list of ideas where the model can be improved.

- Add an image depth analysis module. Indeed, one aspect that is not expressively taken into account in our model is the depth of the different regions present in an image, whereas they should not be treated in the same way. Sometimes the background can be deliberately blurred compared to the foreground and should not be enhanced in the same manner (e.g. a close-up will have a blurred background).
- Reduce the GPU usage of the BasicVSR implementation by breaking the work into different parts. One possibility would be to first compute the backward propagation, then the forward and finish with the upsampling by storing the intermediate values in CPU memory.
- Reduce the GPU usage of the RealBasicVSR implementation by separating the execution of the cleaning module from the BasicVSR part and by storing the cleaned images in between.
- Keep the dynamic refinement and improving it by using a more appropriate metric rather than comparing a residual to a threshold.
- Inspect whether the weights linked to each of the terms in the loss function are still relevant after our training (e.g. λ_{per}).
- Inspect the set of layers K used in the perceptual loss using the VGG19 network and the weights associated with each layer (λ_k in equation 4.19).
- Add a connection between the forward and backward branches within the same level. Thanks to this, the two branches are no longer independent and

the features of each branch are calculated on the basis of information that may come from previous and future images. An illustration is provided in Appendix 8.5.

- Add skip connections between different non-consecutive levels. For example, connect the forward branch of level i to level $i + 1$ but also to level $i + 2$. An illustration is provided in Appendix 8.4.

These last two points are ideas that came to us quite early in the work, but that we did not want to prioritize. Firstly, because we wanted to keep BasicVSR intact and that it is therefore interesting insofar as a complete training can be considered. But above all because these are tracks that had been worked on by Chan et al. in their new models, IconVSR [8] and BasicVSR++ [6].

And finally, of course, it is up to those with more computational capacity to further and better train the variants to more deeply investigate the differences that their implementation implies. So it may also be interesting to implement other variants that are potentially better.

Chapter 7

Conclusion

To conclude, this work attempted to address the problem of using deep learning methods to improve the quality of adventure films. More specifically, it tries to divert the use of super-resolution methods to improve some rushes provided by Agile Birds which suffer from grain and pixelation.

In response to this, a solution was constructed in an innovative fashion. Indeed, the aim was not to reinvent the wheel or to propose solutions that had already been explored.

This was done by starting from strong baselines, first with BasicVSR which aims to reconsider some of the most essential components for VSR guided by three basic functionalities, namely propagation, alignment and upsampling. As a result, BasicVSR achieves significant improvements in speed and quality of restoration over many state-of-the-art algorithms.

The extensibility of BasicVSR has led to the development and thus the analysis of RealBasicVSR, which attempts to solve the challenges posed by the diversity and complexity of real-world VSR degradation. To do this, it proposes a cleaning module that reduces noise and artefacts before propagation, and a stochastic degradation scheme that serves to generalize the model to a higher complexity of deterioration.

From this, a new implementation was proposed with multiple objectives. Firstly, the model had to be further lightened to be able to work on the high resolution images provided by Agile Birds. Secondly, it had to be able to work efficiently on the grain present in the images and thus be able to reduce it. Finally, the contribution had to focus on the restoration of high frequencies in the input images.

From there, a new model was proposed based on a new implementation and the continuation of the training. In view of the results obtained, it can be stated that this model succeeded in achieving the various objectives. Firstly, the model was made even lighter to be able to work on the high resolution images provided by Agile Birds. Secondly, it is able to work efficiently on the grain present in the images and thus to reduce it. Finally, the model had to focus on high frequency restoration and is able to do so on the input images.

Beyond that, this paper can serve as a solid theoretical basis for getting to grips with the subject of video super-resolution. Indeed, it proposes a retrospective analysis of the methods used, develops in detail the techniques considered to be at the cutting edge, through the models that are presented. And finally, it proposes a list of improvements which could guide anyone who would like to take up the progress of this work.

Chapter 8

Appendixes

8.1 Training hyperparameters

	BasicVSR	RealBasicVSR		Our models	
c	64	64	64	64	64
losses	$\mathcal{L}_{out}$	$\mathcal{L}_{out}, \mathcal{L}_{clean}$	$\mathcal{L}_{out}, \mathcal{L}_{clean}$ $\mathcal{L}_{per}, \mathcal{L}_{adv}$	$\mathcal{L}_{out}, \mathcal{L}_{clean}$	$\mathcal{L}_{out}, \mathcal{L}_{clean}$ $\mathcal{L}_{per}, \mathcal{L}_{adv}$
#iteration	300000	300000	150000	10000	2000
B per #GPU	1	2	1	3	1
$(k \in K, \lambda_k)$	/	/	$(2; 0.1), (7; 0.1)$ $(16; 1), (25; 1)$ $(34; 1)$	/	$(2; 0.1), (7; 0.1)$ $(16; 1), (25; 1)$ $(34; 1)$
n_{res}	30	20	20	20	20
n_{clean}	/	20	20	20	20
η_{max}	2e-4	1e-4	5e-5	1e-4	5e-5
θ	/	255	255	255	255

8.2 Flows

8.2.1 BasicVSR

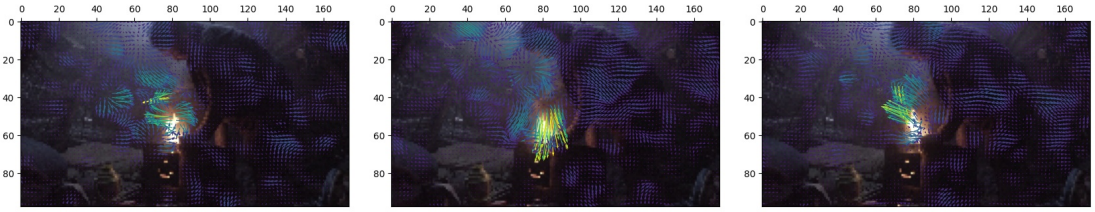


Figure 8.1: Flow of frames index 9, 10, and 11 in BasicVSR, on the video presented on Figure 4.14. Input frames are superimposed and the size of the arrows are 10× longer than the norm of the flow (in pixel distances) for illustration purposes.

8.2.2 RealBasicVSR

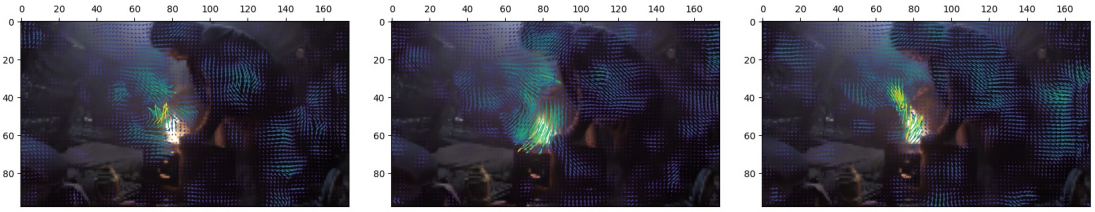


Figure 8.2: Flow of frames index 9, 10, and 11 in RealBasicVSR, on the video presented on Figure 4.14. Input frames are superimposed and the size of the arrows are 10× longer than the norm of the flow (in pixel distances) for illustration purposes.

8.3 Contribution Variants

8.3.1 Cleaning Module of the `4lev_1resPack_withSkip`

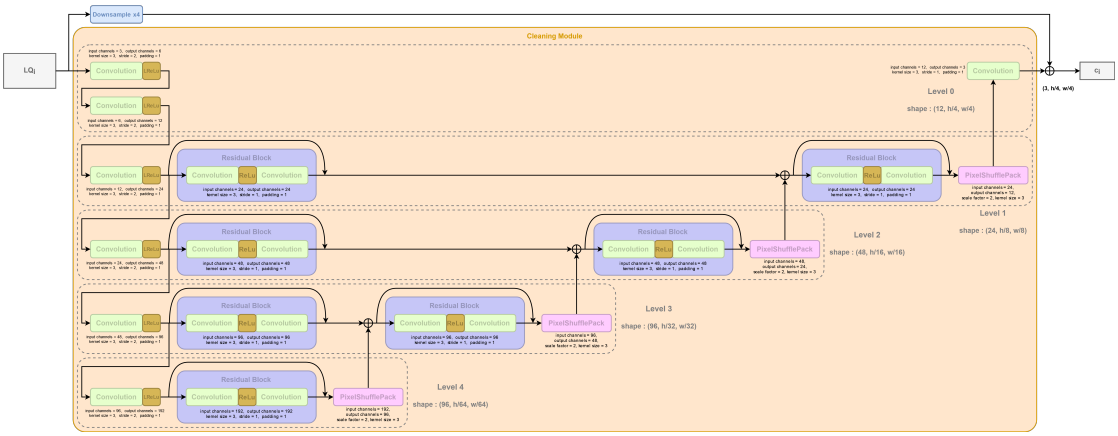


Figure 8.3: Detailed architecture of the cleaning module of the `4lev_1resPack_withSkip` Variant

8.3.2 Cleaning Module of the `2lev_1resPack_withSkip`

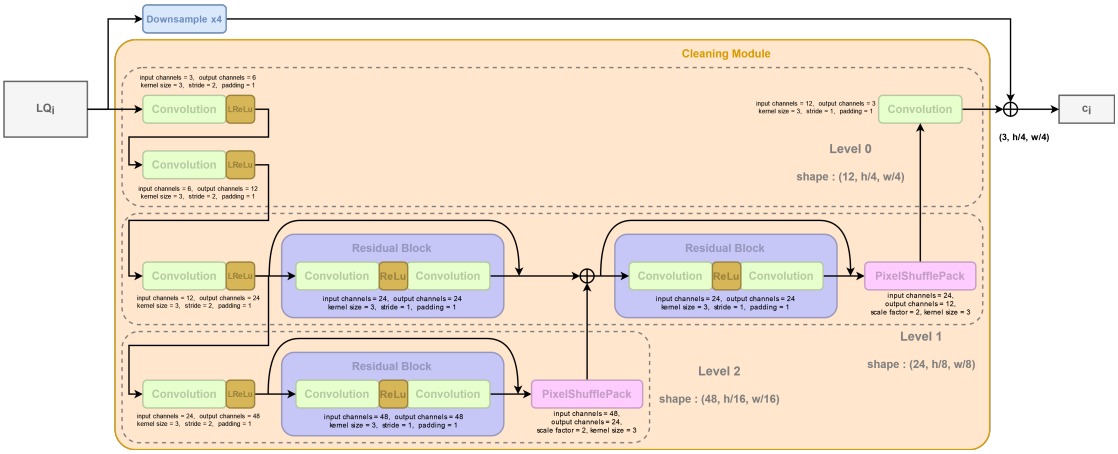


Figure 8.4: Detailed architecture of the cleaning module of the `2lev_1resPack_withSkip` Variant

8.3.3 Cleaning Module of the `3lev_2resPack_withSkip`

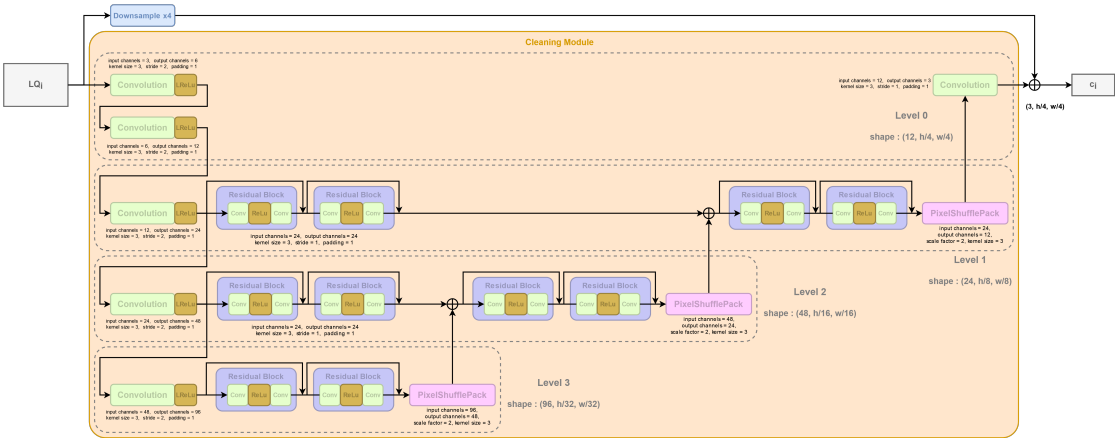


Figure 8.5: Detailed architecture of the cleaning module of the `3lev_2resPack_withSkip` Variant

8.3.4 Cleaning Module of the `3lev_1resPack_woSkip`

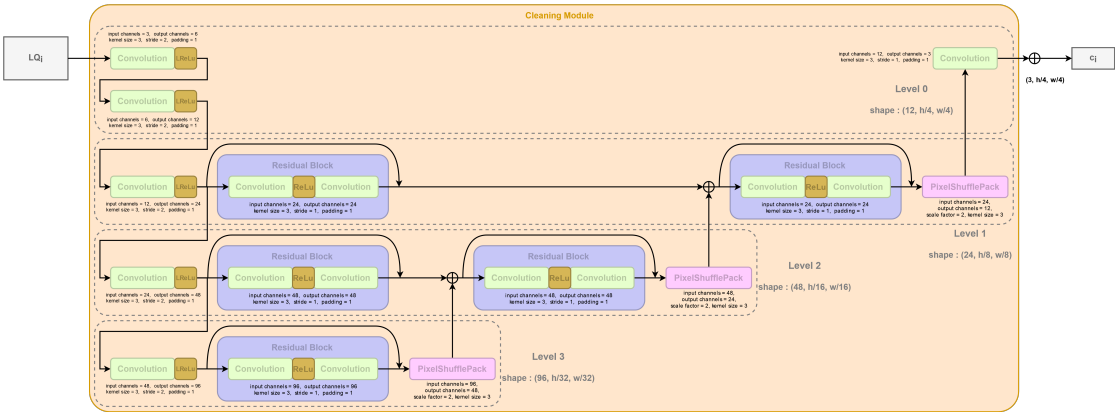


Figure 8.6: Detailed architecture of the cleaning module of the `3lev_1resPack_woSkip` Variant

8.4 Branches Skip Connections

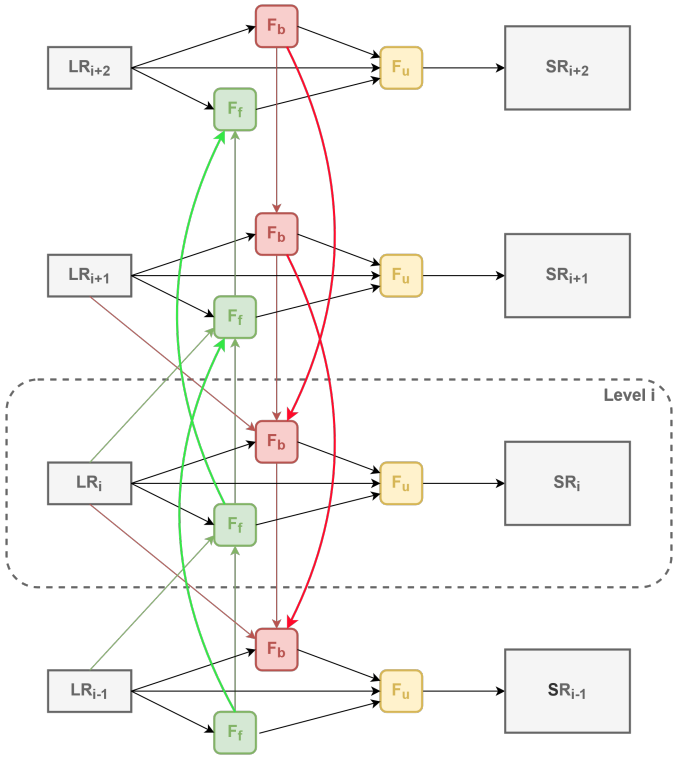


Figure 8.7: Illustration of the improvement of the propagation by adding a skip connection between the levels

8.5 Coupled Propagation

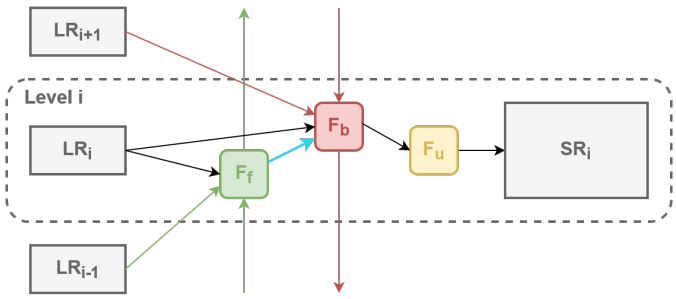


Figure 8.8: Illustration of the improvement of the propagation by coupling the branches

8.6 Variants Results Comparison

Figure 8.9: Comparison of the outputs of the 5 variants

8.7 Final Model Results

Figure 8.10: Results obtained by the final model on the other rushes that had to be improved. Zoom in for more details

Bibliography

[1] Mikołaj Bińkowski, Danica J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. *arXiv preprint arXiv:1801.01401*, 2018.

[2] Yochai Blau, Roey Mechrez, Radu Timofte, Tomer Michaeli, and Lihi Zelnik-Manor. The 2018 pirm challenge on perceptual image super-resolution. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018.

[3] Yochai Blau and Tomer Michaeli. The Perception-Distortion Tradeoff. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6228–6237, June 2018. arXiv: 1711.06077.

[4] Netflix Technology Blog. Toward A Practical Perceptual Video Quality Metric. `https://netflixtechblog.com/toward-a-practical-perceptual-video-quality-metric-653f208b9652`, April 2017.

[5] Aman Chadha, John Britto, and M. Mani Roja. iSeeBetter: Spatio-temporal video super-resolution using recurrent generative back-projection networks. *Computational Visual Media*, 6(3):307–317, September 2020.

[6] Kelvin C. K. Chan, Shangchen Zhou, Xiangyu Xu, and Chen Change Loy. BasicVSR++: Improving Video Super-Resolution with Enhanced Propagation and Alignment, 2021. _eprint: 2104.13371.

[7] Kelvin C.K. Chan. Basicvsr structure why not using a encoder-decoder framework? · discussion #770 · open-mmlab/mmediting.

[8] Kelvin C.K. Chan, Xintao Wang, Ke Yu, Chao Dong, and Chen Change Loy. BasicVSR: The Search for Essential Components in Video Super-Resolution and Beyond. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4945–4954, Nashville, TN, USA, June 2021. IEEE.

[9] Kelvin CK Chan, Shangchen Zhou, Xiangyu Xu, and Chen Change Loy. On the generalization of basicvsr++ to video deblurring and denoising. *arXiv preprint arXiv:2204.05308*, 2022.

[10] Kelvin C.K. Chan, Shangchen Zhou, Xiangyu Xu, and Chen Change Loy. Realbasicvsr: Investigating tradeoffs in real-world video super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2022.

[11] P. Charbonnier, L. Blanc-Feraud, G. Aubert, and M. Barlaud. Two deterministic half-quadratic regularization algorithms for computed imaging. In *Proceedings of 1st International Conference on Image Processing*, volume 2, pages 168–172 vol.2, 1994.

[12] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2015.

[13] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[14] Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. Logan: Evaluating privacy leakage of generative models using generative adversarial networks. *arXiv preprint arXiv:1705.07663*, pages 506–519, 05 2017.

[15] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[16] Image Multimédia Audiovisuel Communication IMAC. Les différentes méthodes d'interpolation. `https://interpolation2016.wordpress.com/methodes-interpolations/`, Mar 2016.

[17] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[18] Tae Hyun Kim, Mehdi SM Sajjadi, Michael Hirsch, and Bernhard Scholkopf. Spatio-temporal transformer network for video restoration. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 106–122, 2018.

[19] Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 624–632, 2017.

[20] Chao Ma, Chih-Yuan Yang, Xiaokang Yang, and Ming-Hsuan Yang. Learning a no-reference quality metric for single-image super-resolution. *Computer Vision and Image Understanding*, 158:1–16, 2017.

[21] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pages 416–423. IEEE, 2001.

[22] Sourav Mishra, Toshihiko Yamasaki, and Hideaki Imaizumi. Improving image classifiers for small datasets by learning rate adaptations. In *2019 16th International Conference on Machine Vision Applications (MVA)*, pages 1–6. IEEE, 2019.

[23] Anish Mittal, Anush Krishna Moorthy, and Alan Conrad Bovik. No-reference image quality assessment in the spatial domain. *IEEE Transactions on Image Processing : a publication of the IEEE Signal Processing Society*, 21(12):4695–4708, 08 2012.

[24] Anish Mittal, Rajiv Soundararajan, and Alan C. Bovik. Making a "completely blind" image quality analyzer. *IEEE Signal Processing Letters*, 20(3):209–212, 2013.

[25] Seungjun Nah, Sungyong Baik, Seokil Hong, Gyeongsik Moon, Sanghyun Son, Radu Timofte, and Kyoung Mu Lee. Ntire 2019 challenge on video deblurring and super-resolution: Dataset and study. In *CVPR Workshops*, June 2019.

[26] Seungjun Nah, Sungyong Baik, Seokil Hong, Gyeongsik Moon, Sanghyun Son, Radu Timofte, and Kyoung Mu Lee. Ntire 2019 challenge on video deblurring and super-resolution: Dataset and study. In *CVPR Workshops*, June 2019.

[27] Mengjiao Qin, Sébastien Mavromatis, Linshu Hu, Feng Zhang, Renyi Liu, Jean Sequeira, and Zhenhong Du. Remote sensing single-image resolution improvement using a deep gradient-aware network with image-specific enhancement.

[28] Anurag Ranjan and Michael J Black. Optical flow estimation using a spatial pyramid network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4161–4170, 2017.

[29] Sabyasachi Sahoo. Residual blocks — building blocks of resnet. https://towardsdatascience.com/residual-blocks-building-blocks-of-resnet-fd90ca15d6ec.

[30] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[31] Xintao Wang, Liangbin Xie, Chao Dong, and Ying Shan. Real-ESRGAN: Training Real-World Blind Super-Resolution with Pure Synthetic Data. In *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 1905–1914, Montreal, BC, Canada, October 2021. IEEE.

[32] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European conference on computer vision (ECCV) workshops*, 2018.

[33] Zhihao Wang, Jian Chen, and Steven C. H. Hoi. Deep learning for image super-resolution: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3365–3387, 2020.

[34] Tianfan Xue, Baian Chen, Jiajun Wu, Donglai Wei, and William T Freeman. Video enhancement with task-oriented flow. *International Journal of Computer Vision*, 127(8):1106–1125, 2019.

[35] Fuzhi Yang, Huan Yang, Jianlong Fu, Hongtao Lu, and Baining Guo. Learning Texture Transformer Network for Image Super-Resolution. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5790–5799, Seattle, WA, USA, June 2020. IEEE.

[36] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 586–595, Salt Lake City, UT, June 2018. IEEE.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**